UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

tesi di laurea

Progetto e Sviluppo di un Sistema per il Tracciamento Sicuro di Oggetti

Anno Accademico 2008/09

relatore
Ch.mo prof. Domenico Cotroneo

correlatore
Ch.mo prof. Marcello Cinque

candidato
Gerardo Daniele Saracino
matr. 534/1280

A Giulia, il mio dolce panzerotto

Indice

Introduzione 6
Capitolo 1 - Applicazioni di Tracciamento e Tecnologia RFID 10
1.1 Tracciamento in Real Time 10
1.2 Tracciamento con Codici a Barre ed RFID 11
1.2.1 Tracciamento con l'utilizzo dei Codici a Barre 12
1.2.2 Tracciamento mediante Tecnologia RFID 13
1.3 L'infrastruttura per il Tracciamento dell'Auto-ID Center 14
1.3.1 Lo Standard EPC 15
1.3.2 La struttura di rete di EPC Global 17
1.4 La Tecnologia RFID 19
1.4.1 I TAG 20
1.4.2 I Reader 25
1.4.3 Frequenze per la comunicazione tra TAG e Reader 26
1.4.4 Classificazione dei TAG per Frequenze e Tecnologie 26
1.4.5 Classificazione dei TAG per Standard e Applicazioni 29
1.4.6 NFC 32
1.5 Sicurezza nei sistemi RFID 33
1.5.1 Tipologie di minacce 33
1.5.2 Crittografia per i sistemi RFID 35
Capitolo 2 - Progettazione di un Sistema di Tracciamento Sicuro 38
2.1 Struttura Generale del Sistema 38
2.2 Inserimento del Percorso sul TAG e Problemi di Sicurezza 41
2.2.1 Inserimento sul TAG del Percorso dell'Oggetto 41
2.2.2 Rilevazione dell'Alterazione delle Informazioni sul TAG 41
2.2.3 Cancellazione e Clonazione dei Dati sul TAG 44
2.2.4 Considerazioni sulle Scelte Adoperate 45
2.3 Casi d'Uso 46
2.4 Diagrammi delle Classi 50
2.5 Scenari d'Utilizzo 52
Capitolo 3 - Un Sistema per il Tracciamento Sicuro delle Merci 55
3.1 Cosa Vogliamo Realizzare 55
3.2 Tecnologie Adottate e Strumenti di Sviluppo 56
3.3 Realizzazione del Sistema 59
3.3.1 I Dati Necessari al Tracciamento 59
3.3.2 Diagramma dei Casi d'Uso 60
3.3.3 Diagrammi delle Classi 61
3.4 Scenari d'Utilizzo 62
Conclusioni e Sviluppi Futuri 67
Appendice A - Tecnologia NFC 69
A.1 Comunicazione tra dispositivi NFC 70
A.2 Standard 71
A.3 NDEF – NFC Data Exchange Format 72
A.3.1 Messaggi NDEF 73
A.3.2 Record NDEF 74

A.3.3 Record Chunks 76
A.4 NFC Record Type Definition 78
A.4.1 NFC Forum well-known type 78
A.4.2 NFC Forum external type 79
A.5 NFC Text Record Type Definition 79
A.6 NFC URI Record Type Definition 81
A.7 NFC Smart Poster Record Type Definition 84
A.7.1 URI Record 85
A.7.2 Title Record 85
A.7.3 Raccomanded Action Record 85
A.7.4 Icon Record 86
A.7.5 Size Record 86
A.7.6 Type Record 86
A.7.7 Esempi di Smart Poster Record 87
Appendice B - La Crittografia 89
B.1 La crittografia simmetrica 91
B.1.1 Il Cifrato di Cesare 91
B.1.2 Il Cifrario di Leon Battista Alberti 92
B.1.3 Il codice di Vigènere 93
B.1.4 Il cifrario a trasposizione per colonne 95
B.1.5 La macchina Enigma 96
B.1.6 Il DES 97
B.1.7 L'algoritmo IDEA 99
B.2 La crittografia asimmetrica 100
B.2.1 L'RSA 101
B.3 Sistemi crittografici ibridi 102
B.4 Funzioni di hash 102
B.5 La firma digitale 104
Appendice C - Java Me, JSR 257 e Boucy Castle Cryptography API 106
C.1 La piattaforma Java 2 Micro Edition 106
C.1.1 Configuration 107
C.1.2 Profile 108
C.1.3 Il CLDC 109
C.1.4 Il MIDP 110
C.2 Le Midlet 112
C.2.1 Struttura di una MIDlet 114
C.2.2 Ciclo di vita di una MIDlet 114
C.2.3 Esempio di MIDlet 117
C.3 Le Contactless Communication API 118
C.4 Le Bouncy Castle Cryptography API 124
Bibliografia 126

Introduzione

Tracciamento rappresenta il termine utilizzato per indicare, specie nel settore della logistica, il processo tramite il quale risulta possibile determinare il percorso seguito da uno o più oggetti in transito da un luogo d'origine verso una destinazione ed, eventualmente, le informazioni inerenti lo stato degli oggetti in questione durante lo spostamento. A supporto di suddetto processo intervengono un insieme di tecnologie che, in base alle loro caratteristiche, permettono di operare differenti modalità di tracciamento. Le tecnologie presenti oggigiorno, infatti, permettono di operare il tracciamento di un oggetto fondamentalmente seguendo due approcci differenti:

- il primo consiste nel monitorare in maniera costante la posizione dell'oggetto lungo il percorso, quindi, in una sorta di modalità Real Time
- il secondo prevede invece la registrazione dell'arrivo e della partenza dell'oggetto nel momento in cui quest'ultimo giunge in una delle tappe in cui il percorso da seguire si articola

Quello di cui noi ci occuperemo nel corso di questo lavoro sarà quello di delineare l'architettura di un sistema di tracciamento che opera sfruttando il secondo approccio. La possibilità di operare il tracciamento oggetti secondo una siffatta modalità comporta però la necessità di avere un modo per identificare, possibilmente in maniera automatica, gli oggetti nel momento in cui giungono in una delle suddette tappe.

Negli ultimi anni è stato possibile assistere ad una crescente diffusione dei sistemi per l'identificazione automatica nel settore terziario al fine di identificare oggetti, animali e persone. Fino ad oggi il sistema per l'identificazione automatica più diffuso è stato il codice a barre che ha portato, dal momento della sua invenzione, ad una vera e propria rivoluzione nel campo dell'identificazione automatica. Il suo principio di funzionamento, semplice ma allo stesso tempo ingegnoso, si basa sulla presenza di una serie di elementi grafici a contrasto elevato su di un etichetta in modo da poter essere letti tramite un sensore a scansione e successivamente decodificati consentendo quindi di ricavare

informazioni. Evidenti limitazioni quali ad esempio l'impossibilità di inserire informazioni aggiuntive sull'etichetta, oltre al codice identificativo rappresentato dal codice a barre, hanno però spinto i tecnici alla ricerca di soluzioni più sofisticate. Ecco quindi comparire nel 1973, in un brevetto statunitense di Mario Cardullo (U.S. Patent 3713148), il primo antenato dell'RFID.

L'RFID rappresenta una tecnologia per l'identificazione automatica che sfrutta le onde radio per leggere e scrivere informazioni su apposite etichette intelligenti denominate TAG. La possibilità di interagire con le etichette senza contatto fisico unita alla maggiore quantità di informazioni che queste ultime sono in grado di immagazzinare hanno aperto le porte a nuovi scenari per l'impiego dei sistemi di identificazione automatica. Particolare rilevanza assumono proprio le capacità di memoria dei TAG. Infatti, mentre in un sistema di identificazione basato sulla tecnologia dei codici a barre risulta infatti unicamente possibile dedurre dall'etichetta un codice identificativo, con l'RFID risulta invece possibile immagazzinare direttamente nell'etichetta i dati inerenti la natura dell'oggetto da identificare.

Le nuove possibilità offerte dalla tecnologia RFID ci permettono quindi di operare il tracciamento oggetti secondo modalità non ipotizzabili in precedenza. Infatti, al giorno d'oggi, il tracciamento oggetti con RFID viene realizzato utilizzando

- TAG a sola lettura sui quali è unicamente presente un numero di serie che permette di identificare univocamente il TAG a livello globale
- una cosiddetta *Unità di Tracciamento* o *Tracker* ovvero un entità che mantiene le informazioni riguardanti il percorso degli oggetti di cui viene tenuta traccia

Quindi, ogni volta che un oggetto giunge in una delle tappe nelle quali il percorso si articola il *Tracker* viene aggiornato con le informazioni riguardanti la posizione attuale dell'oggetto in questione. Sulla base di queste ipotesi, ogniqualvolta ci si trova in presenza di un oggetto e si desidera ottenere le informazioni riguardanti il percorso intrapreso prima di giungere nella sua posizione attuale si deve

1. acquisire il numero di serie caratteristico del TAG attaccato all'oggetto mediante l'utilizzo di un lettore RFID

2. interrogare il *Tracker*, fornendo il numero di serie precedentemente acquisito, il quale potrà quindi comunicare le informazioni di interesse

La memoria con cui sono equipaggiati i moderni TAG unita alla possibilità di poter effettuare scritture multiple ci offrono però la possibilità di individuare la provenienza di un oggetto seguendo un approccio differente: immagazzinare direttamente sul TAG le informazioni sul percorso in modo fa fornire a chiunque sia dotato di un apposito dispositivo di ottenere le informazioni riguardanti la provenienza di un oggetto effettuando semplicemente un operazione di lettura del TAG. In questo modo, ogni volta che un oggetto giunge in una nuova tappa, le informazioni sul percorso non vengono più aggiornate su quello che in precedenza abbiamo chiamato *Tracker* ma direttamente sul TAG. Una simile soluzione non è però immune da problemi legati, in particolar modo, all'impiego di TAG scrivibili. La possibilità, infatti, di poter effettuare sul TAG oltre alle operazioni di lettura anche quelle di scrittura ci pone nelle condizioni di affrontare problemi relativi alla manomissibilità delle informazioni, ossia, relativi al cosiddetto problema del TAG Tampering.

Un potente alleato che, se non in grado di prevenire l'alterazione dei dati sui TAG, ci permette quantomeno di rilevare le eventuali alterazioni è rappresentato dalle moderne tecniche crittografiche ed in particolare dalla crittografia asimmetrica. Basandoci su queste tecniche, abbiamo dunque sviluppato un algoritmo che, mediante la scrittura di determinati valori sul TAG, è in grado di garantire, qualora non ci sia stata contraffazione, che l'oggetto tracciato sia effettivamente passato attraverso la catena di tappe dichiarate sul TAG.

Una classe di dispositivi mobili di prossima generazione in grado di permettere sia la lettura di determinati tipi di TAG che l'esecuzione di algoritmi crittografici, indispensabili al controllo della coerenza dei dati sul TAG, è rappresentata dai telefoni cellulari che integrano al loro interno le tecnologie NFC e Java ME. Le funzionalità offerte dalla tecnologia NFC, la quale può essere considerata una parente prossima dell'RFID, ci hanno, infatti, permesso di realizzare le operazioni di lettura e scrittura del TAG necessarie per il recupero e l'aggiornamento dei dati relativi al percorso dell'oggetto al quale abbiamo

supposto il TAG fosse attaccato.

Sulla base di tutte queste premesse abbiamo organizzato la nostra tesi in

- un Primo Capitolo nel quale viene offerta una breve panoramica su come, al giorno d'oggi, viene realizzato il tracciamento oggetti, con particolare riferimento al tracciamento con RFID
- un Secondo Capitolo nel quale vengono invece esaminate le problematiche riguardanti la progettazione di un sistema di tracciamento basato su tecnologia RFID seguendo l'approccio descritto in precedenza
- un Terzo Capitolo dove vengono descritti gli aspetti inerenti l'implementazione di un sistema di tracciamento merci basato sui concetti esaminati nel precedente capitolo
- una serie di appendici nelle quali vengono descritti gli strumenti dei quali abbiamo dovuto appropriarci per ragionare su quanto discusso nei precedenti capitoli, oltre che per l'implementazione del software

Un ultima nota di merito riguarda le API crittografiche Bouncy Castle tramite le quali è stato possibile implementare l'algoritmo per il check dei valori sul TAG.

Capitolo 1

Applicazioni di Tracciamento e Tecnologia RFID

Il termine tracciamento, specie nel settore della logistica, viene utilizzato per indicare il processo tramite il quale risulta possibile determinare il percorso seguito da uno o più oggetti in transito da un luogo d'origine verso una destinazione. In maniera più rigorosa diremo che il processo di tracciamento di un oggetto consiste nel

- determinarne la posizione geografica corrente
- ricostruirne il percorso compiuto fino alla posizione corrente
- ottenere informazioni inerenti lo stato presente e passato dell'oggetto da tracciare

A supporto di suddetto processo intervengono un insieme di tecnologie che, in base alle loro caratteristiche, permettono di operare differenti modalità di tracciamento. Le tecnologie presenti oggigiorno, infatti, permettono di operare il tracciamento di un oggetto fondamentalmente seguendo due approcci differenti:

- il primo consiste nel monitorare in maniera costante la posizione dell'oggetto lungo il percorso, quindi, in una sorta di modalità Real Time
- il secondo prevede invece la registrazione dell'arrivo e della partenza dell'oggetto nel momento in cui quest'ultimo giunge in una delle tappe in cui il percorso da seguire si articola

Fatte queste premesse, siamo ora in grado di esaminare le modalità con cui le tecnologie per l'identificazione automatica ci vengono in contro per realizzare operazioni di tracciamento oggetti.

1.1 Tracciamento in Real Time

Il tracciamento in Real Time viene oggigiorno realizzato sfruttando la tecnologia GPS (Global Positioning System) tramite la quale è possibile determinare la posizione degli oggetti tracciati ovunque essi si trovino ed in qualunque momento. Infatti, il GPS

rappresenta un Sistema Satellitare Globale di Navigazione ossia un sistema che sfrutta una rete di satelliti artificiali in orbita intorno al pianeta per determinare la posizione del Transponder, costituito in questo caso da un dispositivo con integrato un ricevitore GPS. Per essere più precisi, esistono fondamentalmente due classi di dispositivi che sfruttano la tecnologia GPS per permettere il tracciamento oggetti in modalità Real Time: i Data Pushers e i Data Pullers.

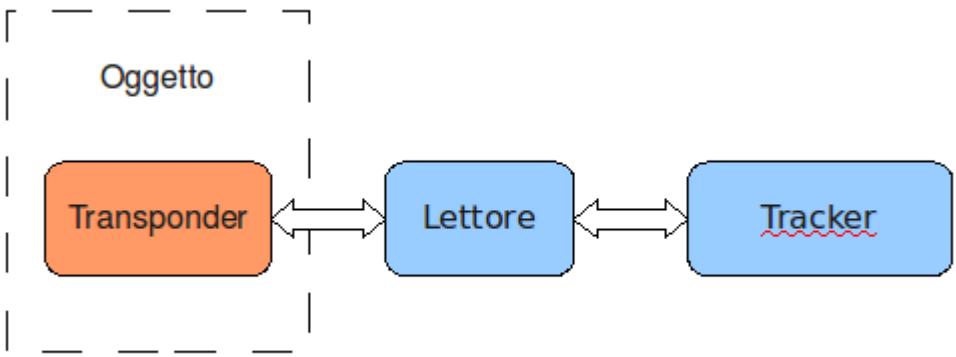
I Data Pushers rappresentano dispositivi che inviano, ad intervalli regolari, le informazioni riguardanti la loro posizione all'Unità di Tracciamento. Si tratta di dispositivi composti da un ricevitore GPS ed una sorta di telefono cellulare il cui compito è quello di inviare all'Unità di Tracciamento, tramite ad esempio messaggi SMS, le informazioni riguardanti la posizione corrente.

Data Pullers i quali, al contrario dei Data Pushers, sono sempre accesi e rispondono alle interrogazioni solo su richiesta da parte del Tracker. Questa classe di dispositivi trova impiego nei casi in cui si renda necessario conoscere soltanto occasionalmente la posizione degli oggetti ai quali vengono attaccati. Infatti, è possibile impiegare questo genere di dispositivi per il tracciamento della posizione di beni soggetti a furto come le auto.

1.2 Tracciamento con Codici a Barre ed RFID

Le tecnologie dei codici a barre e quella RFID consentono entrambe di operare il tracciamento di un oggetto seguendo un approccio che prevede la registrazione dell'arrivo e della partenza dell'oggetto nel momento in cui quest'ultimo giunge in una delle tappe in cui il percorso da seguire si articola. È possibile immaginare una tappa come un punto intermedio all'interno del percorso che l'oggetto da tracciare compie dal suo punto d'origine fino alla destinazione. Il motivo per cui il tracciamento viene operato in un simile scenario è da ricercarsi nella natura stessa dei codici a barre e dell'RFID le quali rappresentano entrambe tecnologie per l'identificazione automatica. L'Identificazione Automatica rappresenta il generico termine che consente di classificare un insieme di tecnologie per l'identificazione degli oggetti tramite l'ausilio delle macchine. Mediante

queste tecnologie, risulta possibile catturare informazioni sugli oggetti ed inviarle ad un sistema di elaborazione, rappresentato nel nostro caso dall'Unità di Tracciamento, in maniera automatica cioè senza che nessuno si occupi di inserire manualmente le informazioni nel sistema di elaborazione. Lo scopo con cui sono stati concepite queste tecnologie è quello di incrementare l'efficienza e ridurre gli errori dovuti all'intervento umano nell'inserimento dei dati nei sistemi di elaborazione. La lettura dell'elemento transponder viene realizzata tramite l'ausilio di appositi dispositivi lettori che a loro volta si interfacciano con l'unità di tracciamento vera e propria.



Interazione tra Tracker, Lettore e Transponder

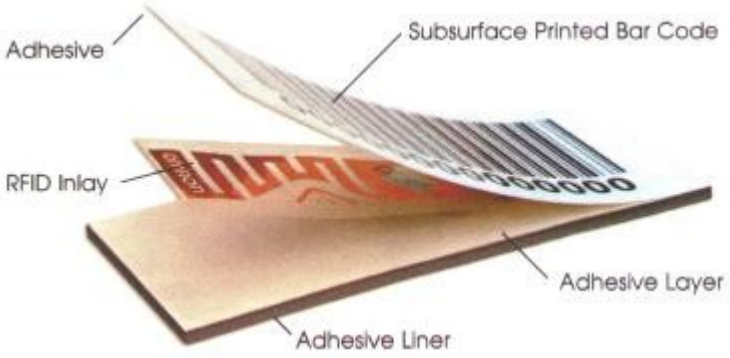
1.2.1 Tracciamento con l'utilizzo dei Codici a Barre

I codici a barre sono un insieme di elementi grafici a contrasto elevato disposti in modo da poter essere letti da un apposito lettore e decodificati per restituire l'informazione contenuta che consiste generalmente in un codice identificativo. I maggiori svantaggi nell'utilizzo di questa tecnologia risiedono nella necessità che l'etichetta, sulla quale è stampato il codice a barre vero e proprio, risulti visibile.

Esempio di Codice a Barre

1.2.2 Tracciamento mediante Tecnologia RFID

L'RFID (Radio Frequency Identification) rappresenta una tecnologia per l'identificazione, attraverso trasmissioni a radiofrequenza, di oggetti, animali e persone. L'identificazione avviene mediante l'utilizzo di apposite etichette RFID denominate anche TAG in grado di trattenere informazioni sugli oggetti a cui sono associate. Alla fine del capitolo esamineremo più nel dettaglio questa tecnologia ma per ora ci basta sapere che esistono diversi metodi per identificare gli oggetti tramite l'RFID ma il più comune è quello di inserire all'interno del TAG un numero di serie che identifica univocamente l'oggetto a cui il TAG è associato. Suddetto numero di serie, una volta acquisito dal lettore, può essere quindi inviato all'Unità di Tracciamento.



Struttura di un etichetta RFID Passiva

L'identificazione degli oggetti attraverso trasmissioni a radiofrequenza porta con se diversi vantaggi rispetto a quella con i codici a barre. Riportiamo di seguito nella seguente tabella un sunto dei maggiori vantaggi a cui porta l'utilizzo della tecnologia RFID rispetto a quella dei codici a barre.

CODICI A BARRE	RFID
Necessità che il lettore veda il codice da leggere	Per la lettura del TAG non è necessaria alcuna visibilità
Il lettore può leggere un solo codice alla volta	Il Reader è in grado di effettuare più letture simultaneamente
Per la lettura di una serie di etichette viene impiegato un certo tempo	La lettura simultanea di una serie di TAG impiega pressappoco lo stesso tempo necessario alla lettura di uno solo
Le etichette devono rimanere pulite e non devono deformarsi	La lettura avviene anche in presenza di materiali ricoprenti il TAG

1.3 L'infrastruttura per il Tracciamento dell'Auto-ID Center

L'Auto-Id Center è un progetto di ricerca diretto e coordinato dal MIT (Massachusetts Institute of Technology), in collaborazione con altri cinque noti centri accademici a livello mondiale (Cambridge, Adelaide, Keio, Fudan, S. Gallen) e sponsorizzato da numerose e prestigiose multinazionali nei più svariati settori come Gillette, Coca-Cola, Wal-Mart, Johnson & Johnson, Carrefour, Tesco, Pepsi, Protect & Gamble, UPS, Kraft, Philips Morris oltre che organismi come Ean International, Uniform Code Council e da svariati fornitori di tecnologia quali Philips Semiconductor, Siemens, Symbol, Intermec, Alien, Zebra, Avery, SAP, IBM, Matrics, Sun Microsystems. Il progetto costituisce una joint venture tra mondo accademico e industriale finalizzato allo studio e allo sviluppo di nuovi standard per le tecnologie RFID e l'utilizzo dei TAG per l'identificazione univoca, su scala mondiale, dei singoli oggetti, facilitando gli scambi commerciali, lo scambio delle informazioni e le operazioni di gestione delle merci nella filiera produttiva e distributiva, la cosiddetta supply chain.

Il 31 Ottobre 2003 l'Auto-Id Center termina ufficialmente la sua missione consegnando in gestione i nuovi standard (EPC, ONS, Savant, PML ecc.) e i risultati delle ricerche effettuate a EPC Global, un nuovo ente no-profit costituito da EAN International e UCC, e presieduto da un collegio di quindici membri, per la maggior parte rappresentanti scelti tra i più importanti sponsor del progetto Auto Id Center, il quale invece continua la sua opera

di ricerca accademica sulle tecnologie RFID sotto il nuovo nome di Auto-Id Labs.

1.3.1 Lo Standard EPC

Scopo primario dell'Auto-Id Center era la definizione di un nuovo standard di codifica, adottabile su scala mondiale, per qualsiasi prodotto o contenitore, basato sulla codifica di un TAG RFID da apporre su dì esso. Tale nuovo standard è costituito dal codice EPC, l'Electronic Product Code, del quale esistono due versioni: la versione *short* a 64 bit e la *standard* a 96 bit. Suddetto standard prevede uno schema di codifica internazionale che permette di identificare univocamente un singolo prodotto, l'item, oppure un contenitore, l'SKU ovvero Stock Keeping Unit, distinguendolo da qualsiasi altro esemplare uguale presente sul mercato. Di seguito illustriamo, mediante un esempio, la struttura di un EPC a 96 bit.

EPC a 96 bit			
Header (8 bit)	**EPC Manager (28 bit)**	**Object Class (24 bit)**	**Serial Number (36 bit)**
21	203D2A9	16E8B8	719BAE03C

Struttura di un codice EPC a 96 bit

Possiamo notare come in suddetto EPC siano presenti diversi campi:

- **Header** il quale descrive la struttura che segue ed il tipo di EPC
- **EPC Manager** il quale rappresenta un codice univoco ed identificativo del produttore del prodotto sul quale è applicato il TAG
- **Object Class** ovvero un codice univoco ed identificativo di una certa tipologia di oggetti
- **Serial Number** ossia il numero di serie del particolare oggetto

Il principale elemento di novità rispetto agli attuali schemi adottati da organismi quali l'UCC, Uniform Code Council, o il suo corrispettivo europeo EAN, European Article Numbering, è la presenza del numero seriale, in aggiunta al codice del produttore, l'EPC Manager, e del prodotto stesso, l'Object Class.

Questo elemento di novità, che deriva dalle maggiori possibilità tecnologiche rispetto al codice a barre offerte dalla codifica in chip di memoria, permetterà, al contrario di oggi, di distinguere ad esempio due bottiglie identiche di Cola Cola da 33 cl, potendo risalire facilmente a luogo, data e lotto di produzione, locazione attuale, e a molte altre informazioni utili.

Poiché lo standard EPC prevede di identificare unicamente ogni singolo articolo , il che significa l'apposizione di un TAG ad ogni oggetto esistente, è evidente che ciò implica la produzione e l'utilizzo di miliardi di TAG negli anni a venire, il che è economicamente pensabile soltanto qualora il costo dei TAG si riduca ad una soglia di accessibilità economica paragonabile a quella di una normale etichetta con codice a barre, soglia che per il momento è stata valutata in 5 centesimi di dollaro contro gli attuali 25 centesimi e più necessari all'acquisto di un TAG RFID con limitate capacità di memoria. Risulta quindi evidente che in tale situazione è necessario contenere al massimo il costo del TAG per permettere l'adozione su larga scala del codice EPC e la conseguente riduzione dei prezzi dei TAG. Per fare ciò, essendo il costo del TAG direttamente proporzionale alla quantità di memoria, è particolarmente importante memorizzare nel chip solo l'informazione strettamente necessaria a identificare l'oggetto, potendo memorizzare qualsiasi ulteriore informazione ad esso associata in un database. Per tale motivo, nel periodo transitorio, l'Auto-Id Center ha proposto e previsto un EPC ridotto a 64 bit, più che sufficiente per le attuali esigenze, per contenere il costo dei TAG e innescare più

velocemente il fenomeno della riduzione dei prezzi su larga scala.

1.3.2 La struttura di rete di EPC Global

L'approccio individuato dall'Auto-Id Center è quello di creare una infrastruttura software di collegamento tra gli oggetti fisici identificati tramite il codice EPC, e le informazioni che li descrivono, memorizzate su database pubblici, creando un sistema globale basato su standard aperti, e utilizzando la tecnologia di Internet per rendere accessibile in ogni momento, e in ogni luogo, le informazioni su qualsiasi prodotto.
I componenti essenziali dell'architettura EPC Global sono:

- il codice EPC memorizzato sul TAG
- il Reader per la lettura del TAG
- il Savant ovvero un middleware che si occupa di trattare opportunamente i dati ricevuti dai lettori, di interagire con le applicazioni aziendali legacy, di gestione della supply chain e con altri Savant
- un server ONS, Object Name Service, preposto a restituire l'indirizzo di un server pubblico contenente le informazioni sul prodotto. Possiamo immaginare che svolga un ruolo analogo a quello dei server DNS
- Un server PML pubblico che memorizza tutti i dati del prodotto in un apposito linguaggio PML, Pysical Markup Language, basato sullo standard XML

I lettori, posti ad esempio sulle porte di entrata / uscita delle merci, nei camion, negli scaffali dei punti vendita, acquisiscono simultaneamente i numerosi codici EPC dai prodotti che si trovano nel proprio raggio d'azione, gestendo opportunamente, tramite appositi protocolli di trasmissione, le possibili collisioni tra i lettori stessi e tra i TAG, e le possibili mancate letture.
Il flusso di codici risultante viene inviato a un apposito software di gestione, il Savant, che si occupa di trattare opportunamente i dati scartando le doppie letture, e prendendo le necessarie decisioni su quali dati passare ai software ai quali è collegato. Un Savant recupera e abbina ai codici le informazioni sui prodotti interrogando opportunamente un server PML, individuato tramite il sistema ONS.

La funzione di Savant nel sistema è particolarmente critica in quanto si occupa della gestione della comunicazione con i lettori, della ricezione e del trattamento dei dati, evita il sovraccarico della rete mantenendo un database locale real-time basato su eventi (RIED), e assume tutte le decisioni su come trattare i dati in base un modello di gestione TMS (Task Management System) programmabile dall'utente.

Lo standard di Savant è stato concepito sulla base di un'architettura gerarchica distribuita, quindi esisteranno differenti software Savant dislocati nei magazzini, nei centri di distribuzione, nelle fabbriche, nei punti vendita, e persino nei camion e nei cargo, in pratica ovunque siano installati dei lettori RFID.

I vari Savant, come già accennato in precedenza, dialogano tra loro. Ad esempio, il Savant in un magazzino può informare il Savant del centro di distribuzione che un certo prodotto scarseggia, quindi deve essere spedito al più presto. Quindi il Savant del centro di distribuzione può rispondere comunicando che sarà inviata una spedizione di una certa quantità del prodotto a un certa data e una certa ora.

Per recuperare i dati sul prodotto identificato da un codice EPC, il Savant interroga un server ONS posto sulla rete locale o su Internet per recuperare l'indirizzo di un server PML che contiene le informazioni sul prodotto specifico, similmente al meccanismo DNS oggi utilizzato in Internet per risolvere un nome di host in un indirizzo IP.

I dati sui prodotti sono memorizzati in appositi server pubblici mantenuti dai produttori dei prodotti, utilizzando un linguaggio chiamato PML (Pysical Markup Language) e basato sul noto standard XML. Questi dati possono essere di vario genere ovvero

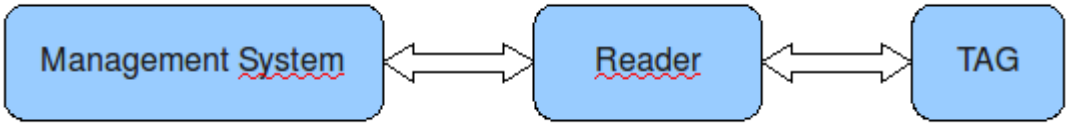
- di tipo costante
- dinamici
- di tipo variabile

I dati di tipo costante potranno ad esempio essere la descrizione del prodotto, le date di confezionamento e di scadenza, composizione ad altro ancora. I dati di tipo variabile potranno riguardare invece ad esempio la temperatura di un macchinario mentre infine quelli di tipo variabile saranno invece quelli inerenti la posizione dell'oggetto all'interno della supply chain.

1.4 La Tecnologia RFID

L'RFID (Radio Frequency Identification) rappresenta una tecnologia per l'identificazione, attraverso trasmissioni a radiofrequenza, di oggetti, animali e persone. L'identificazione avviene mediante l'utilizzo di apposite etichette RFID denominate anche TAG o Transponder in grado di trattenere informazioni sugli oggetti a cui sono associate e ad opera di particolari dispositivi, denominati Reader, in grado di leggere le informazioni che i TAG sono in grado di fornire. Esistono diversi metodi per identificare gli oggetti tramite l'RFID ma il più comune è quello di inserire all'interno del TAG un numero di serie che identifica univocamente l'oggetto a cui il TAG è associato. Suddetto numero di serie, una volta acquisito dal Reader, può essere quindi inviato ad un sistema di elaborazione che se ne servirà per compiere le elaborazioni utili alla particolare logica applicativa. Riassumendo in maniera formale quanto abbiamo appena detto, un sistema per l'identificazione automatica basato su tecnologia RFID si può considerare strutturato in tre componenti fondamentali:

- **TAG** che memorizza le informazioni sull'oggetto a cui è associato. Come vedremo più avanti ne esistono di diversi tipi ma si presentano sempre dotati di un chip e di un antenna.
- **Reader** ossia il dispositivo in grado di leggere ed eventualmente anche scrivere le informazioni in un TAG.
- **Sistema di Elaborazione**, connesso al Reader, in grado di ricavare, a partire da i codici presenti nei TAG, le informazioni utili alla particolare logica applicativa. Infatti la quantità di memoria con cui sono equipaggiati i TAG può essere modesta e tale da contenere unicamente un codice identificativo univoco. Il Sistema di Elaborazione utilizza quindi questo codice identificativo come chiave di ricerca in una base dati per ricavare le informazioni utili all'elaborazione. Risulta importante notare che, quello che noi qui abbiamo chiamato genericamente Sistema di Elaborazione, coincide, nell'ambito di un sistema di tracciamento con l'Unità di Tracciamento.



Interazione tra TAG, Reader e Sistema di Elaborazione

1.4.1 I TAG

I TAG rappresentano il cuore della tecnologia RFID. Si tratta di dispositivi che, a seconda dell'oggetto di cui devono fornire identificazione, possono assumere dimensioni diverse ed implementare funzionalità diverse. Quindi il TAG, nel caso si voglia identificare un prodotto da banco di un supermercato assumerà forma di etichetta adesiva, nel caso si voglia identificare una persona potrà ad esempio essere integrato in un portachiavi e cosi via. La possibilità di leggere informazioni dai TAG è solo una delle tante offerte da questa tecnologia infatti particolari TAG possono presentare particolari caratteristiche. Possono dunque

- essere **read only** ossia fornire unicamente la funzionalità di lettura
- essere **write once** quindi dare la possibilità di inserire al loro interno informazioni una sola volta
- essere **write many** quindi avere possibilità di scritture multiple
- implementare meccanismi **anti-collision**, indispensabili affinché un Reader sia in grado di effettuare la lettura simultanea di più TAG
- implementare tecniche di **security and encryption** tramite le quali la comunicazione col Reader può avvenire in forma cifrata

Oltre che in base alle caratteristiche che presentano, i TAG possono essere classificati a seconda delle modalità di gestione delle fonti energetiche. La capacità che queste etichette intelligenti hanno di trasmettere informazioni al Reader presuppone infatti l'esistenza di una fonte energetica in grado di alimentare i circuiti trasmettitori.

I TAG possono pertanto essere distinti in

- **Passivi** quando non sono dotati di un sistema autonomo di alimentazione. L'energia

necessaria alla trasmissione delle informazioni viene ricavata dal segnale proveniente dal Reader.

- **Semi-passivi** dotati di una batteria interna necessaria soltanto per l'alimentazione di apparati ausiliari quali ad esempio sensori interni. La trasmissione dati avviene con le medesime modalità con cui avviene nei TAG passivi.
- **Attivi** dotati di batteria, incorporano al loro interno un ricevitore ed un trasmettitore, esattamente come i Reader.

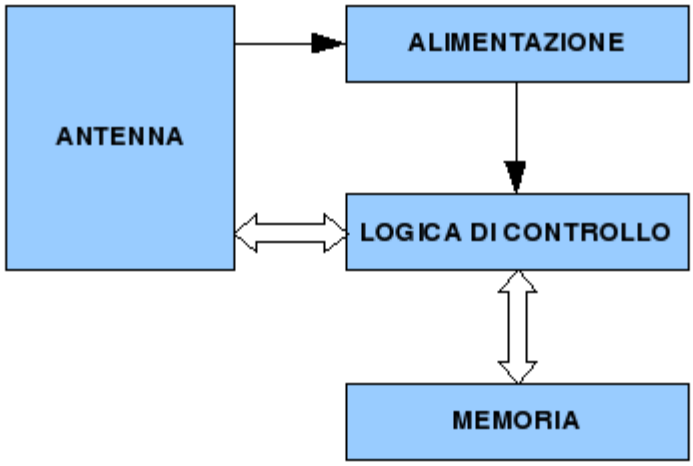
I TAG Passivi rappresentano una tipologia di dispositivi caratterizzata dalle dimensioni che possono essere ridotte anche in maniera estrema e tali da permetterne l'integrazione anche in oggetti di piccole dimensioni quali etichette adesive, carte di credito, bottoni, banconote, biglietti dell'autobus, ecc.. Come spiegato in precedenza sono caratterizzati dall'assenza di un sistema di alimentazione autonomo, quindi di una batteria interna, ed usano il segnale emesso dal Reader per ricavare l'energia necessaria sia ad alimentare la circuiteria interna che a trasmettere. In particolare l'energia proveniente dal segnale esterno, raccolta dall'antenna, viene raddrizzata e portata in un condensatore che funge da batteria consentendo quindi di effettuare la trasmissione. La potenza ricavabile da un segnale esterno è però molto bassa da cui conseguono distanze operative modeste (al massimo di qualche metro). Inoltre le capacità di memoria di questa tipologia di TAG è in genere modesta (è ridotta a pochi Kbyte che servono a memorizzare un codice identificatore dell'oggetto associato e poche altre informazioni aggiuntive). Tuttavia quelle che da un lato possono rappresentare delle limitazioni contribuiscono d'altro canto ad abbassare i costi di produzione di questi dispositivi permettendone quindi la diffusione. Riguardo alla struttura interna un TAG RFID Passivo si presenta costituito da un Antenna ed un Chip. L'antenna in genere stampata e realizzata di materiale conduttore ha il compito di trasdurre il campo elettromagnetico captato in segnale elettrico e viceversa mentre all'interno del chip sono integrate tutte le funzioni indispensabili al funzionamento del TAG. In particolare all'interno di quest'ultimo vengono implementate

- **Alimentazione** atta a rendere funzionante il dispositivo, si occupa di convertire in continua l'energia proveniente dall'antenna e di accumularla in un condensatore che

svolge poi il ruolo di batteria

- **Logica di Controllo** tramite la quale vengono coordinate le operazioni di accesso alla memoria
- **Memoria** costituita generalmente da una EEPROM ed in grado di immagazzinare al limite qualche Kbyte

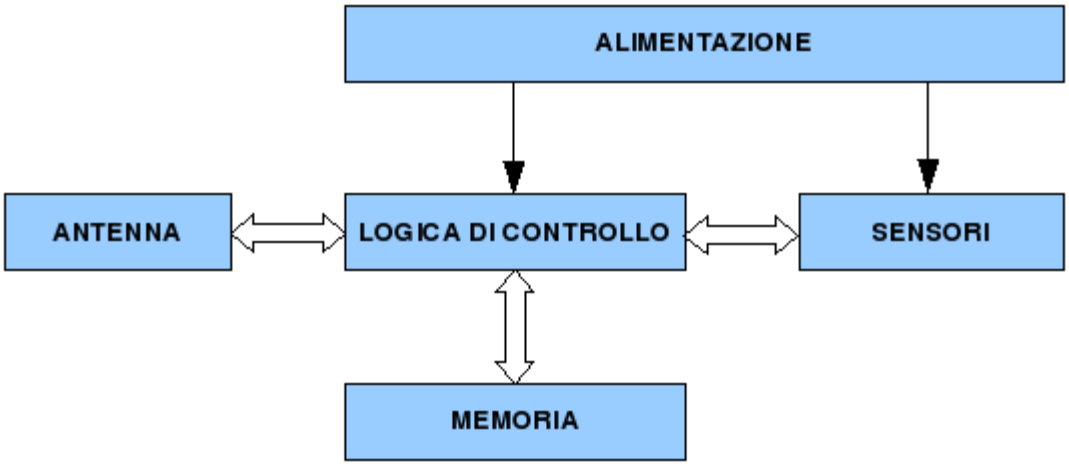
Di seguito viene riportato uno schema che illustra l'interazione tra le diverse componenti di un TAG Passivo.



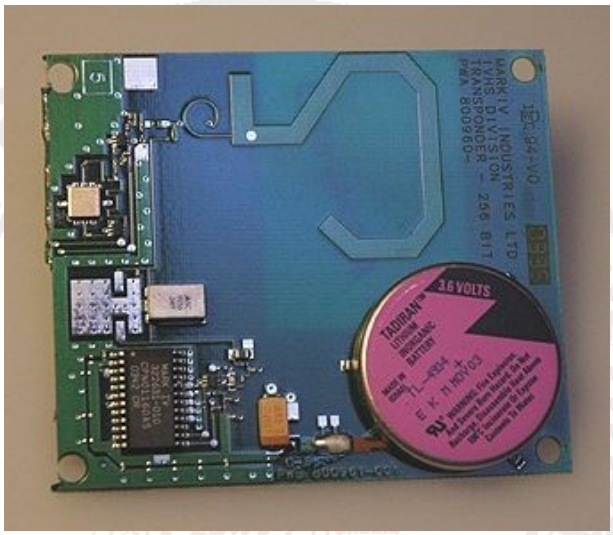
Schema di un TAG Passivo

Esattamente come i TAG passivi, quelli Semi-passivi sfruttano il segnale proveniente dal Reader per effettuare la trasmissione ma a differenza di questi ultimi integrano una batteria indispensabile ad alimentare i circuiti necessari allo svolgimento di operazioni peculiari del particolare dispositivo. La presenza di un'alimentazione indipendente consente a questo genere di dispositivi di svolgere operazioni più complesse rispetto ai loro colleghi completamente passivi, di gestire quantità di memoria più elevate e di effettuare su quest'ultima sia operazioni di lettura che di scritture multiple. Inoltre risulta possibile montare su di essi sensori ambientali, di posizione ed altro ancora. Tuttavia le modalità con cui vengono trasmesse le informazioni ne limitano ancora una volta le distanze operative che comunque risultano superiori a quelle dei TAG Passivi, essendo, in questo caso, il segnale del Reader necessario unicamente alla trasmissione dati e non

all'alimentazione del TAG. Infine proprio la batteria con i suoi costi e la sua durata costituisce il principale punto critico di questi dispositivi. Per ovviare a questi problemi alcuni TAG restano infatti in una sorta di stato di stand by fin quando non vengono risvegliati da un Reader tramite un segnale. Riportiamo di seguito un possibile schema rappresentativo del funzionamento di un TAG Semi-passivo.



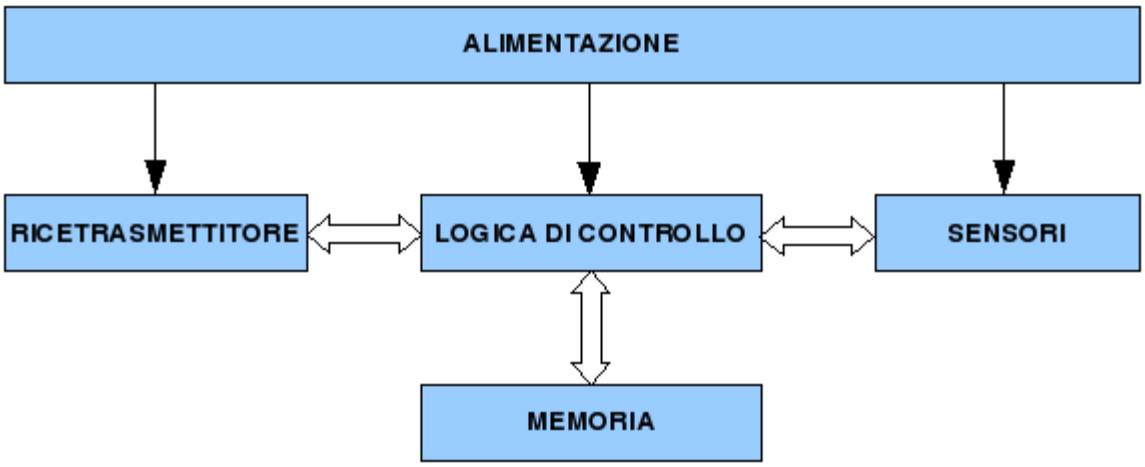
Schema di un TAG Semi-passivo



TAG RFID Semi-passivo

I TAG Attivi, esattamente come i TAG Semi-passivi, incorporano una batteria indispensabile allo svolgimento delle particolari funzioni per cui sono stati concepiti. La marcia in più di questo tipo di TAG risiede però nella presenza di un vero e proprio

ricetrasmettitore che consente il raggiungimento di distanze in trasmissione maggiori (possono arrivare anche a qualche Chilometro).



Schema di un TAG Attivo

TAG RFID Attivo

Infine proponiamo, di seguito in tabella, un confronto tra i diversi tipi di TAG di cui abbiamo poc'anzi parlato con lo scopo di mettere in evidenza quelli che sono i vantaggi e gli svantaggi nell'impiego di ciascuna tipologia.

	VANTAGGI	SVANTAGGI	OSSERVAZIONI
PASSIVI	• Durata elevata • Dimensioni contenute • Basso costo • Meccanicamente flessibili	• Distanza operativa limitata a 4-5 metri • Emissioni Radio controllate rigorosamente da regolamentazioni locali	• Rappresenta la tipologia maggiormente utilizzata di TAG RFID • Bande: LF, HF e UHF
SEMI-PASSIVI	• In grado di controllare sensori • Distanze operative maggiori rispetto ai TAG RFID Passivi	• Costi più elevati causati dalla presenza della batteria e dei sensori • Problemi di affidabilità legati al ciclo di vita limitato della batteria	• Utilizzati principalmente in sistemi in tempo reale per rintracciare materiali di alto valore • Bande: UHF
ATTIVI	• Distanza operativa superiore a quella degli altri TAG • In grado di controllare sensori • Non rientrano nelle rigorose normative per i TAG Passivi		• Usati nella logistica per il tracciamento di container su treni e camion • Bande: UHF

1.4.2 I Reader

Il Reader, denominato anche Interrogator o Controller se separato dalla sua antenna, rappresenta il componente, di un sistema RFID, che consente la lettura delle informazioni detenute dal TAG.

I Reader per TAG Attivi sono dei ricetrasmettitori controllati che possono sfruttare diverse tecniche di radiofrequenza. Un discorso a parte va invece fatto riguardo ai Reader per TAG Passivi e Semi-passivi. Infatti in questo caso deve essere emesso un segnale in

radiofrequenza particolare tale da fornire al TAG anche l'energia per trasmettere.

Un ultima distinzione riguarda quella tra Reader fissi, come quelli montati nelle uscite dei magazzini di distribuzione prodotti (denominati Gate Reader), e Reader portatili, simili, come forma, ai lettori di codici a barre.

1.4.3 Frequenze per la comunicazione tra TAG e Reader

Le frequenze di comunicazione tra TAG e Reader sono scelte in funzione dell'applicazione prevista e, al fine di evitare interferenze nei segnali radio, controllate dagli organismi internazionali e locali di standardizzazione.

Sulla base dell'applicazione prevista frequenze più o meno alte sono scelte a seconda

- delle distanze alle quali si vuol far avvenire la comunicazione
- della velocità di trasferimento dati
- delle dimensioni che si vuole abbia l'antenna ed, in generale, il TAG

Per piccole distanze vengono spesso utilizzati sistemi a TAG Passivi che operano in bassa frequenza. Viceversa TAG Attivi, i quali lavorano a frequenze più elevate, sono impiegati quando si vogliono coprire distanze maggiori. Inoltre al crescere della frequenza operativa si ha un aumento della velocità di trasferimento dati ed una diminuzione delle dimensioni dell'antenna.

Gli standard che regolamentano le frequenze in uso dai sistemi RFID variano nei vari paesi. Risulta comunque possibile distinguere i seguenti range di frequenze più comunemente usati:

- **LF** (Low Frequencies) ed in particolare la sottobanda 120÷145 KHz
- **HF** (High Frequencies) ed in particolare la sottobanda centrata sui 13,56 MHz
- **UHF** (Ultra High Frequencies) ed in particolare le sottobande 865÷870 Mhz in uso in Europa e 902÷928 Mhz in USA
- **SHF** (Super High Frequencies) ed in particolare la sottobanda centrata sui 2,4GHz

1.4.4 Classificazione dei TAG per Frequenze e Tecnologie

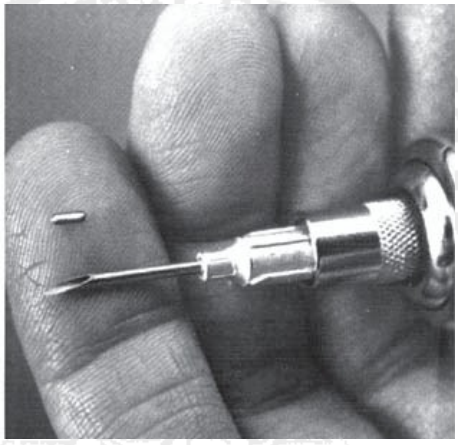
Servendoci di quanto appreso fin ora andremo a fornire una descrizione di varie tipologie

di TAG in relazione alle bande di frequenza per la comunicazione avendo cura di mettere in evidenza le tecnologie impiegate. Distingueremo pertanto

- TAG INDUTTIVI LF
- TAG INDUTTIVI HF
- TAG ELETTROMAGNETICI UHF
- TAG ELETTROMAGNETICI SHF

I TAG operanti nella banda LF costituiscono una tipologia di TAG, generalmente passivi, nei quali l'accoppiamento col Reader avviene per via induttiva (Inductive Coupling) e sono caratterizzati dalla velocità di trasmissione dati relativamente bassa (può scendere in alcuni casi fino a 200 bit/s), dalle basse distanze operative (massimo mezzo metro) e dall'impossibilità, in molti casi, di effettuare letture multiple. La banda di frequenza nella quale operano conferisce loro la capacità di trasmettere anche in presenza di liquidi e metalli. Va inoltre notato che all'interno della banda LF le frequenze portanti per lo più in uso sono

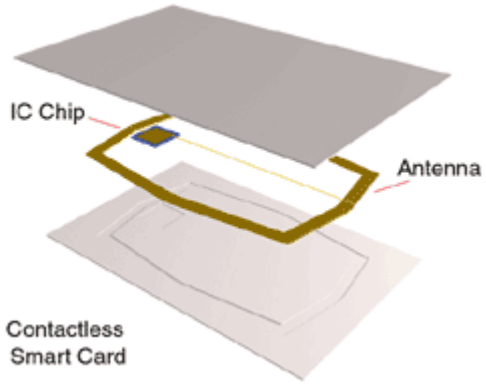
- 125,5 KHz nel settore dell'automazione industriale
- 134,2 KHz per la tracciabilità animale



TAG LF Induttivo da iniettare per via sottocutanea

Anche i TAG operanti in banda HF sono generalmente passivi ed esattamente come i TAG LF, sfruttano l'accoppiamento induttivo. Inoltre, presentano la più ampia diffusione a livello mondiale, dovuta soprattutto alla banda in cui operano, libera da vincoli normativi. Molto spesso implementano meccanismi anticollisione, il che si traduce nella possibilità,

del Reader, di effettuare letture multiple. Inoltre presentano memorie con capacità che vanno dal Kilobyte fino ad arrivare al Megabyte con possibilità sia di lettura che di scrittura. La velocità di trasmissione dati è bassa ma comunque superiore a quella dei TAG LF (si aggira intorno ai 25Kbit/s) mentre la distanza operativa arriva intorno al metro. Transponder di questo tipo vengono spesso impiegati nella produzione delle Smart Card Contactless, le cosiddette carte elettroniche senza contatto.



Struttura di una Smart Card Contactless

I TAG UHF, invece, sfruttano l'accoppiamento elettromagnetico e sono in genere attivi. La presenza di un vero e proprio apparato ricetrasmettitore con cui sono equipaggiati questi dispositivi si traduce in distanze operative più elevate e velocità di trasmissione dati maggiori rispetto a quelle di altri TAG visti fin ora. L'alimentazione autonoma inoltre consente loro di integrare memorie dalla capacità più ampia e con possibilità sia di lettura che scrittura. Tuttavia circostanze di ordine sia normativo che tecnico ne hanno fin ora rallentato la diffusione. Le normative per le gestione dei segnali radio su banda UHF, che risulta di per se già affollata, infatti variano da paese a paese. C'è da notare infine che a queste frequenze la trasmissione diventa difficoltosa in presenza di liquidi e metalli.

Infine, i TAG SHF sono molto simili, data la loro alta frequenza operativa, ai TAG UHF e si scontrano quindi con problematiche dello stesso ordine. Possiamo notare infine che la banda sulla quale agiscono consente una consistente diminuzione delle dimensioni dell'antenna.

La seguente tabella illustra le differenze tra i vari TAG descritti fin ora.

	TAG LF	TAG HF	TAG UHF	TAG SHF
Bande di frequenza	120÷145KHz	Centrata su 13,56MHz	865÷870MHz in EU 902÷928MHz in USA	Centrata su 2,4GHz
Accoppiamento	Induttivo	Induttivo	Elettromagnetico	Elettromagnetico
Alimentazione	Passivi	Quasi sempre passivi	Sia Passivi che Attivi	Sia Passivi che Attivi
Distanza Operativa	Tipicamente < 0,5m	Intorno a 1m	Tipicamente 2÷5m	Tipicamente 1÷3m
Capacità di Lettura/Scrittura	Sia R/O che R/W	Sia R/O che R/W	Tipicamente R/W ma anche R/O	Sia R/O che R/W
Velocità di Trasferimento Dati	La più Bassa	Bassa	Media	Alta
Capacità di lettura in presenza di liquidi e/o metalli	La Migliore	Media	Bassa	Molto Bassa
Applicazioni tipiche	Controllo accessi, tracciamento animali, immobilizzazione veicoli	Smart cards, controllo accessi, tracciamento di prodotti e oggetti inclusi bagagli (tranne in USA), librerie, videostore, ecc..	Tracciamento di pallets e contenitori, sistemi di pedaggio elettronico, gestione bagagli (USA)	Supply chain management, sistemi di pedaggio elettronico

1.4.5 Classificazione dei TAG per Standard e Applicazioni

Proponiamo di seguito la descrizione di settori applicativi nei quali vengono utilizzati dispositivi basati su RFID avendo cura di mettere in evidenza gli standard adoperati.

I transponder per l'identificazione di animali aderiscono a tre standard che prevedono

frequenze operative al di sotto dei 135KHz. Questi standard sono ISO 11784, ISO 11785 ed ISO 14223. Era previsto inizialmente un blocco di 64 bit necessario a contenere unicamente un identificatore ma successivamente con l'introduzione dell'ISO 14223, l'ultimo introdotto, si è provveduto a definire blocchi di dati riscrivibili o protetti da scrittura.

L'insieme degli standard inerenti il settore della logistica sono definiti da due differenti organismi: ISO ed EPCglobal. In particolare gli standard ISO sono per lo più orientati alla definizione delle interfacce radio mentre quelli EPCglobal riguardano anche i settori applicativi. La serie di standard 18000, proposta ISO, definisce i meccanismi anticollisione, l'interfaccia radio ed i protocolli di comunicazione a differenti bande di frequenza per una serie di TAG destinati alla logistica e all'identificazione di oggetti. Suddetti standard sono suddivisi per parti ed in particolare

- **Parte 1** che provvede a definire l'architettura di riferimento
- **Parti da 2 a 7** la quale definisce caratteristiche peculiari di sistemi operanti a frequenze differenti ed in particolare
  - **Parte 2**, definisce TAG operanti a frequenza inferiori ai 135 KHz
  - **Parte 3** nella quale sono definite due sottoparti: 3-1 e 3-2. Le due sottoparti definiscono le caratteristiche di sistemi operanti intorno ai 13,56MHz ed in particolare la seconda quelli con velocità di trasferimento dati più alta
  - **Parte 4**, definizione di sistemi con banda intorno ai 2,45GHz ed in dettaglio TAG a **modo 1**, passivi ed in backscatter e a **modo 2**, attivi e con alto data rate
  - **Parte 6**, sistemi passivi con TAG in backscatter sui 900MHz
  - **Parte 7** nella quale è definito un sistema RFID a lungo raggio con TAG attivi nella banda dei 433MHz

EPCglobal, invece, oltre agli standard strettamente inerenti i livelli applicativi, ha definito standard riguardanti diverse tipologie di TAG:

- **Classe 0 – Generation 1**, operanti a 900MHz, passivi ed a sola lettura, attualmente non più utilizzati. Il blocco dati a bordo di questi TAG è costituito da un codice

identificatore (denominato EPC) definito in sede di fabbricazione e non alterabile. Nella classe 0 è inoltre definita una particolare tipologia di TAG denominati EAS (Electronic Article Surveillance) i quali non sono dotati neanche di un identificatore ma consentono la loro rilevazione una volta nei pressi del Reader.

- **Classe 1 – Generation 1**, con portante a 13,56 o 900MHz, passivi e con possibilità di un unica scrittura. Sono, come quelli definiti nella precedente classe, considerati obsoleti.
- **Classe 1 – Generation 2**, a 900MHz, passivi e scrivibili una sola volta, ampliano le caratteristiche definite per le precedenti classi.

Le carte elettroniche senza contatto sono particolari tipi di transponder che operano in banda HF. Rappresentano un evoluzione delle comuni smart card definite mediante le specifiche ISO/IEC 7810. L'ISO, in particolare, definisce tre standard per questo tipo di dispositivi:

- **Close-coupled cards**, definite nell'ISO/IEC 10536, sono dispositivi che operano a distanza dal Reader inferiore ad 1 cm. Si tratta di dispositivi per cui si sta gradualmente interesse data la diffusione che stanno avendo quelli descritti di seguito
- **Proximity cards**, nell'ISO/IEC 14443, che operano a distanza dal Reader inferiore a 10 cm. Inoltre sono dotate di microprocessore, quindi presentano una logica di controllo molto evoluta. Esistono due tipi di smart card contactless definite dallo standard che differiscono per la modulazione utilizzata: Tipo A e Tipo B. Le prime quattro parti dello standard descrivono le caratteristiche fisiche, il tipo di modulazione (tipo A o B), l'inizializzazione e l'anticollisione ed infine il protocollo di trasmissione. I protocolli di trasmissione più utilizzati sono due ed entrambi di tipo proprietario: il MIFARE di Philips o il FeliCa sviluppato da Sony.
- **Vicinity Cards**, descritte dall'ISO/IEC 15693, sono in grado di operare a distanze pari al metro e mezzo. Presentano una logica di controllo più semplice di quella implementata nelle precedenti vengono generalmente impiegate in applicazioni come il controllo degli accessi. Lo standard descrive in particolare l'interfaccia

radio, il meccanismo di anticollisione ed il protocollo di trasmissione.

1.4.6 NFC

Si tratta di una tecnologia per la connettività in radiofrequenza con distanza operativa inferiore ai 10cm, banda intorno ai 13,56MHz e con velocità di trasferimento dati che possono raggiungere i 424Kbit/s. L'NFC rappresenta una tecnologia compatibile con gli standard ISO 18092, ISO 14443 e ISO 15693 e nasce a mezzo di un iniziativa di gruppi industriali per creare un ponte tra terminali mobili di larga diffusione e tecnologie contactless. In particolar modo, come abbiamo già avuto modo di vedere, gli standard ISO 14443 e 15693 definiscono le specifiche rispettivamente inerenti le proximity e le vicinity cards mentre l'ISO 18092, denominato anche protocollo NFCIP-1, definisce una modalità operativa peculiare dei dispositivi che implementano NFC. Infatti ciò che contraddistingue l'NFC dall'RFID, nella quale il TAG è unicamente in grado di rispondere alle richieste del Reader, è il fatto che la comunicazione può avvenire come in un sistema peer to peer nel quale ciascuna delle entità coinvolte ha la possibilità di decidere quali informazioni richiedere alla sua controparte. In particolare quando un dispositivo NFC, l'Initiator, intende iniziare una transazione verso il suo corrispondente, il Target, lo interroga esattamente come avviene in un comune sistema RFID nel quale è il Reader ad interrogare il TAG. Da questo punto in poi, però, la comunicazione può continuare sia in modalità Attivo-Passivo che Attivo-Attivo.

EPC ed NFC rappresentano tecnologie che devono le loro differenze ai differenti scopi per cui sono state concepite infatti mentre NFC rappresenta un approccio per l'integrazione degli standard delle smartcard all'interno di dispositivi mobili di largo consumo EPC nasce dall'esigenza di gestire in maniera efficiente le unità logistiche all'interno della supply chain. Nonostante queste differenze, però, entrambe le tecnologie si propongono come obbiettivo quello di recuperare informazioni circa gli oggetti sui quali è stato apposto un TAG RFID. Quindi introducendo nei dispositivi mobili con tecnologia NFC la compatibilità con l'EPC si potrebbe ad esempio consentire ai possessori di suddetti dispositivi di accedere alla rete EPC semplicemente avvicinando il dispositivo all'oggetto

munito di TAG EPC. Tuttavia, al momento, questo non sembra auspicabile dato che le due tecnologie sono tra di loro incompatibili infatti mentre i TAG compatibili con NFC operano in banda HF e sfruttano l'accoppiamento induttivo, i TAG compatibili con EPC attualmente in uso, quelli di Classe 1 – Generation 2, operano invece in banda UHF e si basano sull'accoppiamento elettromagnetico anche se è in previsione la definizione di uno standard per TAG EPC del tipo Classe 1 – Generation 2 operanti in banda HF. In particolare se lo standard emergente per i TAG EPC in banda HF sarà costruito sulla base delle specifiche ISO 15693 si potrebbe, tramite un semplice aggiornamento del firmware, rendere compatibili i dispositivi mobili NFC con questa nuova tipologia di TAG.

1.5 Sicurezza nei sistemi RFID

Come tutte le tecnologie dell'informazione anche l'RFID non è rimasto a lungo immune dalle tecniche fraudolente per l'acquisizione e l'alterazione delle informazioni. La ricerca di soluzioni atte a prevenire la messa in pratica di suddette tecniche richiede come punto di partenza una definizione rigorosa delle tipologie di attacco, denominate Threath, che è possibile mettere in pratica sui sistemi RFID.

Quello che faremo in questo paragrafo sarà di spiegare brevemente le più diffuse tipologie di attacco che risulta possibile mettere in pratica sui sistemi RFID strizzando, in particolar modo, l'occhio a quelle che mirano all'alterazione diretta dei dati presenti sul TAG (Attacchi di tipo TAG Tampering). Ci occuperemo successivamente di spiegare in che modo le moderne tecniche di crittografia possono venirci in contro ed aiutarci al fine di combattere suddetto tipo di minacce.

1.5.1 Tipologie di minacce

Riportiamo quindi nella seguente tabella un elenco delle principali tipologie di attacchi a cui sono suscettibili i sistemi RFID.

Principali modelli di Threat	
Denominazione generale	Modalità di attacco
Sniffing	Intercettazioni (Eavesdropping)
	Letture non autorizzate dei TAG
TAG Tampering	
Data spoofing	Data Corruption
	Data modification
	Data insertion
	Man in the middle attack
Malware	
Denial o service	

In linea del tutto generale, è possibile definire lo sniffing come l'attività di intercettazione dei dati in transito attraverso un canale di comunicazione. Nel nostro caso, essendo lo scambio dei dati tra TAG e Reader rappresentato da una comunicazione in radiofrequenza, risulta evidente la possibilità di mettere in pratica minacce di questo tipo. Minacce di questo tipo possono essere attuate tramite l'utilizzo di un ricevitore e di un antenna durante la comunicazione tra il TAG ed il Reader. La soluzione ad un simile problema consisterà innanzitutto quindi nell'impedire l'intercettazione della comunicazione da parte di un terzo estraneo. Un secondo aspetto da tenere in considerazione è rappresentato dalla decodifica, da parte dell'estraneo, delle informazioni ad intercettazione avvenuta. Mentre, per quanto riguarda il primo problema non è possibile dare una risposta univoca per quanto riguarda il secondo possiamo intervenire mediante tutta una serie di note tecniche di cifratura al fine di rendere sicuro il canale di comunicazione.

Gli attacchi di tipo Data Spoofing mirano generalmente alla modifica dei dati in transito durante la comunicazione tra TAG e Reader. A seconda delle intenzioni dell'estraneo che mette in pratica l'attacco è possibile parlare di

- **Data Corruption**, attacco nel quale l'aggressore tenta di modificare i dati in transito in modo che vengano ricevuti alterati. L'alterazione però, è intesa solamente come disturbo della comunicazione, non essendo l'aggressore in grado di interpretare i dati ed alterarli in modo da farli apparire coerenti

- **Data Modification** con la quale, come nel caso precedente, l'aggressore cerca di alterare i dati in transito ma con lo scopo che una volta ricevuti vengano ritenuto validi
- **Data Insertion** nel quale l'estraneo introduce messaggi apparentemente corretti tra i dati scambiati tra TAG e Reader. La messa in pratica di un simile intervento richiede però che gli intervalli con i quali vengono scambiati i messaggi tra i due apparati siano sufficientemente lunghi e tali da permettere l'iniezione dei dati estranei
- **Man in the Middle Attack** nel quale i due apparati coinvolti nella comunicazione, il TAG ed il Reader, vengono ingannati da un terzo estraneo e coinvolti in una conversazione a tre. In particolare, nel corso di quest'attacco il TAG e il Reader non comunicano direttamente tra di loro ma con l'intruso che utilizza i dati provenienti dagli altri due apparati al fine di renderli incoscienti dell'intrusione

Il termine TAG Tampering fa riferimento ad una tipologia di attacco che mira all'alterazione diretta dei dati presenti sul TAG. A differenza delle tipologie di attacco viste fin ora, nelle quali veniva messa in pratica l'alterazione della comunicazione tra TAG e Reader, il TAG Tampering viene generalmente messo in pratica tramite l'ausilio di Reader portatili tramite i quali alterare direttamente i dati sul TAG. La possibilità di eseguire suddetto tipo di attacchi è strettamente legata alla tipologia di protezione con cui le informazioni vengono scritte sul TAG. Il TAG Tampering è infatti favorito sia nel caso in cui non venga implementata alcuna forma di autenticazione per l'accesso in scrittura al TAG (ad esempio tramite password), sia nel caso in cui le informazioni sul TAG vengano riportate in chiaro. Le opportune limitazioni nell'accesso al TAG unite all'applicazione di opportuni strumenti crittografici costituiscono i mezzi per contrastare efficacemente questo tipo di minacce.

1.5.2 Crittografia per i sistemi RFID

L'obbiettivo principale dell'applicazione di strumenti crittografici nei sistemi RFID è quello di proteggere dalla manomissione i dati memorizzati sul TAG. Premettiamo subito

che, specie per i TAG economici, quelli passivi e destinati alla grande distribuzione, è necessario trovare modi sostenibili per utilizzare gli strumenti di crittografia. Infatti l'utilizzo di tecniche crittografiche efficaci richiederebbe una potenza di calcolo nei chip dei TAG che non è disponibile nei modelli più economici. Pertanto le tecniche più robuste che risulta possibile utilizzare sono quelle che non richiedono elevate capacità di calcolo sul TAG ma solamente sul Reader.

Più formalmente possiamo dire che, tramite l'utilizzo della crittografia, vogliamo evitare che soggetti non autorizzati possano

- alterare i dati presenti sul TAG
- ottenere l'accesso per leggere i dati sul TAG
- clonare il TAG

Con lo scopo di impedire che i dati memorizzati sul TAG possano essere alterati o quantomeno per consentire una validazione dei dati presenti sul TAG vengono utilizzati strumenti come la firma digitale. Ci occuperemo di spiegare in maniera dettagliata in cosa consiste la firma digitale in appendice ma per ora ci basti sapere che tramite questo strumento risulta possibile verificare che i dati non siano stati alterati.

Con il fine di proteggere invece il TAG da eventuali letture non autorizzate ed impedire, di conseguenza, che il TAG possa essere clonato vengono impiegate tecniche, tramite le quali, il Reader effettua un autenticazione nei confronti del TAG prima di procedere con la lettura. Vengono prevalentemente impiegate due modalità con cui è possibile effettuare l'autenticazione:

- Mutual Symmetrical Keys
- Derived Keys

La Mutual Symmetrical Keys è una tecnica di autenticazione descritta dallo standard ISO 9798-2 che prevede che le due controparti coinvolte nella comunicazione, il TAG e il Reader, verifichino reciprocamente la conoscenza di una chiave segreta. Questo protocollo prevede che il Reader, denominato in questo contesto *Claimant*, invii al TAG, il *Verifier*, un messaggio contenente un valore pseudocasuale. Il TAG, da parte sua, risponde con un messaggio che dipende dal valore contenuto nel messaggio ricevuto precedentemente e

dalla chiave segreta condivisa. Una volta avvenuta suddetta fase di autenticazione la comunicazione può iniziare in forma cifrata, sfruttando la chiave condivisa di cui le controparti sono a conoscenza.

La modalità di autenticazione Derived Keys permette di derivare una chiave caratteristica di ogni coppia di apparati che entrano in comunicazione. Suddetta chiave viene in genere ricavata a partire dal numero di serie di uno dei sue apparati, il TAG o il Reader, che si suppone sia unico a livello mondiale e da un valore segreto, condiviso dalle due entità. In questa modalità il Reader richiede al TAG il suo numero di serie e attraverso quest'ultimo e il valore segreto condiviso genera una chiave per cifrare la comunicazione.

Capitolo 2

Progettazione di un Sistema di Tracciamento Sicuro

Dopo aver offerto, nel precedente capitolo, una panoramica su come, sfruttando alcune delle tecnologie per l'identificazione automatica disponibili oggigiorno, risulti possibile realizzare operazioni di tracciamento oggetti e su come in particolare la tecnologia RFID possa essere sfruttata per realizzare suddetta operazione procederemo ora con la definizione dell'architettura generale di un sistema per il tracciamento sicuro di oggetti. Mentre, come già spiegato nel precedente capitolo, il termine tracciamento identifica il generico processo finalizzato al raggiungimento di obbiettivi quali

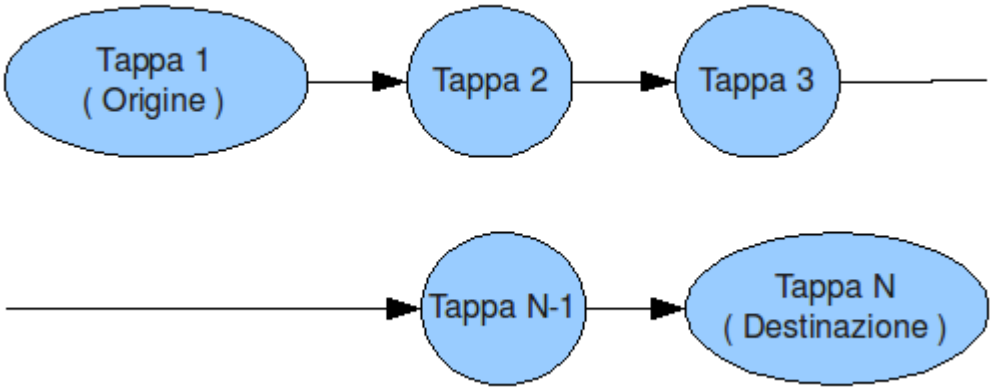
- ottenere informazioni circa il luogo in cui si trova un determinato oggetto
- ottenere informazioni circa il percorso intrapreso dall'oggetto da tracciare prima di giungere nella sua posizione attuale
- ottenere informazioni inerenti lo stato presente e passato dell'oggetto in questione

noi focalizzeremo l'attenzione sullo sviluppo di un sistema che permetta di raggiungere soltanto i primi due dei tre obbiettivi sopra citati operando, quindi, unicamente il tracciamento della posizione di un oggetto. L'architettura di suddetto sistema sarà basata sull'impiego della tecnologia RFID come mezzo per identificare gli oggetti di cui tener traccia. Il nostro lavoro però non si fermerà qui, infatti, ci occuperemo di proporre delle soluzioni a quelli che sono i problemi di sicurezza che l'impiego di suddetta tecnologia introduce.

2.1 Struttura Generale del Sistema

Come spiegato nel precedente capitolo, l'impiego dell'RFID per applicazioni di tracciamento permette di operare seguendo un approccio che prevede la registrazione dell'arrivo e della partenza dell'oggetto nel momento in cui quest'ultimo giunge in una delle tappe in cui il percorso da seguire si articola. Suddetto percorso, dunque, può

considerarsi strutturato come mostrato di seguito.



Il percorso di un Oggetto in un Sistema di Tracciamento basato su RFID

Le informazioni inerenti il tragitto compiuto dall'oggetto e quindi le informazioni sulle diverse tappe attraverso le quali l'oggetto è passato sono mantenute dall'*Unità di Tracciamento* o *Tracker*. Quindi, se opportunamente interrogato, il *Tracker* è in grado di fornire agli interessati suddette informazioni. Ogni volta che un oggetto giunge in una determinata tappa le informazioni mantenute dal *Tracker*, inerenti le tappe attraverso le quali l'oggetto è passato, vengono aggiornate.

Affinché il *Tracker* sia in grado di determinare il luogo in cui un determinato oggetto si trova in un certo momento deve essere in grado innanzitutto di identificare la specifica tappa del percorso. A ciascuna tappa, quindi, sarà associato un identificatore univoco che chiameremo d'ora in poi ID_TAPPA il quale viene generato dal *Tracker* all'atto della procedura di registrazione della tappa.

L'identificazione automatica di un oggetto in ciascuna tappa avviene invece tramite la lettura di un identificatore univoco dell'oggetto memorizzato sul TAG il cui valore viene stabilito dal *Tracker* all'atto della richiesta di tracciamento la quale avviene nella tappa iniziale del percorso, quella che nel precedente schema è stata identificata come l'origine dell'oggetto. Denoteremo d'ora in l'identificatore dell'oggetto con il termine ID_OGGETTO. Va detto che il valore di suddetto identificatore univoco dell'oggetto, l'ID_OGGETTO, può essere determinato dal *Tracker*

- nel caso vengano impiegati TAG a sola lettura, acquisendo semplicemente il numero seriale già presente sul TAG
- nel caso di utilizzo di TAG riscrivibili, generandone magari uno ad hoc

In maniera più rigorosa, le informazioni riguardanti le tappe in cui il percorso di un oggetto può articolarsi e quelle relative alle tappe percorse da un oggetto vengono gestite dal *Tracker* tramite la consultazione di due tabelle che d'ora in avanti chiameremo rispettivamente *Tabella Tappe* e *Tabella Oggetti*.

Tabella Tappe	
ID_TAPPA 1	Informazioni per l'identificazione della tappa in questione
...	...
ID_TAPPA N	...

Tabella Oggetti	
ID_OGGETTO 1	ID_TAPPA 1 ID_TAPPA 2 … ID_TAPPA N
…	…
ID_OGGETTO M	...

Tipico esempio di un sistema di tracciamento basato su di un architettura simile a quella che abbiamo illustrato è rappresentato dai sistemi di tracciamento dei corrieri espresso. Proprio come nell'architettura che abbiamo iniziato a delineare nei sistemi di tracciamento dei corrieri, infatti, la merce compie un percorso, dal suo luogo di provenienza fino a giungere a destinazione, che si articola in una serie di magazzini intermedi, quelle che abbiamo fin ora identificato come le tappe del percorso. Anche in questo caso ogni volta che la merce spedita giunge in uno dei suddetti magazzini viene informata l'*Unità di Tracciamento*. Tutto questo si traduce nella possibilità che ci viene fornita dai corrieri di monitorare la posizione delle spedizioni da noi effettuate tipicamente tramite un interfaccia web.

2.2 Inserimento del Percorso sul TAG e Problemi di Sicurezza

Come è stato già accennato nell'introduzione di questa tesi, il nostro scopo è quello di sfruttare la memoria con cui sono equipaggiati i moderni TAG unitamente alla possibilità di poter effettuare scritture multiple al fine di riuscire ad operare il tracciamento oggetti seguendo un approccio differente: immagazzinare direttamente sul TAG le informazioni sul percorso.

2.2.1 Inserimento sul TAG del Percorso dell'Oggetto

Operando secondo un simile approccio, ogni volta che un oggetto giunge in una nuova tappa, le informazioni sul percorso non vengono più aggiornate solamente sul *Tracker*, al fine di consentire un monitoraggio da remoto della posizione e del percorso seguito, ma anche sul TAG. Quindi sul TAG deve essere inserita una tabella simile alla seguente nella quale viene aggiunto un campo ID_TAPPA ogni volta che l'oggetto di cui viene tenuta traccia dal *Tracker* giunge in una nuova tappa.

ID_OGGETTO
ID_TAPPA 1
…
ID_TAPPA N

Va detto, inoltre, che un simile approccio permette di rilevare, senza che sia necessario consultare il *Tracker*, il percorso compiuto dall'oggetto di interesse effettuando semplicemente la lettura del TAG ed eliminando, quindi, l'onere da parte del *Tracker* di fornire le informazioni sul percorso compiuto dall'oggetto. Tuttavia, come abbiamo avuto modo di vedere in precedenza, i dati presenti sul TAG possono essere facilmente alterati. Occorre, perciò, introdurre un modo per rilevare anche in questo caso l'eventuale manomissione delle informazioni sul TAG.

2.2.2 Rilevazione dell'Alterazione delle Informazioni sul TAG

Una soluzione può essere quella di inserire all'interno del TAG un ulteriore campo che consenta di rilevare le eventuali manomissioni e a cui d'ora in poi faremo riferimento

come al VALORE_DI_CHECK. Sul TAG dovranno essere quindi inseriti i dati presenti nella seguente tabella.

ID_OGGETTO
ID_TAPPA 1
...
ID_TAPPA N
VALORE_DI_CHECK

Il metodo da noi scelto per il calcolo del VALORE_DI_CHECK si basa sull'utilizzo della crittografia asimmetrica. In particolare a ciascuna delle tappe nelle quali può articolarsi il percorso che un oggetto compie viene associata una coppia di chiavi: una chiave pubblica, una cui copia viene detenuta dall'*Unità di Tracciamento* ed una chiave privata, la cui conoscenza è riservata agli operatori della specifica tappa. L'elenco delle chiavi pubbliche associate alle rispettive tappe può essere può essere gestita dal *Tracker* tramite una variante di quella che in precedenza abbiamo chiamato *Tabella Tappe* la quale assumerà una forma simile alla seguente.

Tabella Tappe		
ID_TAPPA 1	PUBLIC_KEY 1	Informazioni per l'identificazione della tappa in questione
...	...	...
ID_TAPPA N	PUBLIC_KEY N	...

Fatta questa premessa siamo finalmente in grado di descrivere in che modo avviene la creazione e l'aggiornamento del VALORE_DI_CHECK. Abbiamo precedentemente detto che il *Tracker*, nel momento in cui gli viene inoltrata la richiesta di tracciamento di un oggetto, quando quest'ultimo si trova nel suo luogo d'origine, risponde generando un ID_OGGETTO e comunicandolo alla tappa dalla quale è partita la richiesta. In suddetta tappa si provvederà ad inserirlo sul TAG, unitamente all'ID_TAPPA della tappa in questione, da attaccare all'oggetto. Sempre presso la tappa dalla quale è partita la richiesta

di tracciamento viene generato il VALORE_DI_CHECK. Questo valore sarà inizialmente costituito dalla cifratura, mediante la chiave privata associata alla tappa in questione, dell'ID_OGGETTO. La procedura di aggiornamento del VALORE_DI_CHECK risulta del tutto analoga a quella della sua creazione. Quando infatti l'oggetto giunge nella successiva tappa del percorso, dopo aver informato il *Tracker* dell'arrivo dell'oggetto, il VALORE_DI_CHECK viene determinato come la cifratura, mediante la chiave privata associata alla tappa attuale, del VALORE_DI_CHECK presente in precedenza sul TAG. Lo stesso discorso può ripetersi per quanto riguarda le tappe successive. In maniera più formale il processo di creazione e aggiornamento del VALORE_DI_CHECK può essere illustrato dal seguente algoritmo:

1. Nella prima tappa, il luogo d'origine dell'oggetto, il VALORE_DI_CHECK viene determinato come la cifratura, mediante la chiave privata della prima tappa, dell'ID_OGGETTO
2. Nella seconda tappa il VALORE_DI_CHECK viene determinato come la cifratura, mediante la chiave privata della seconda tappa, del VALORE_DI_CHECK già presente sul TAG
3. Nella terza tappa il VALORE_DI_CHECK viene determinato come la cifratura, mediante la chiave privata della terza tappa, del VALORE_DI_CHECK già presente sul TAG
4. …
5. …

In base a quanto detto, quindi, l'algoritmo per il controllo dell'autenticità delle informazioni sul TAG si articolerà nei seguenti passaggi:

1. Viene recuperata presso il *Tracker* la PUBLIC_KEY corrispondente all'n-esima tappa identificata sul TAG con ID_TAPPA N
2. Viene decifrato il VALORE_DI_CHECK con la PUBLIC_KEY ottenuta al passo precedente ottenendo un valore che chiameremo VALORE_DI_CHECK N-1
3. Viene recuperata presso il *Tracker* la PUBLIC_KEY corrispondente all'(n-1)-esima tappa identificata sul TAG con ID_TAPPA N-1

4. Viene decifrato il VALORE_DI_CHECK N-1 con la PUBLIC_KEY ottenuta al passo precedente ottenendo un valore che chiameremo VALORE_DI_CHECK N-2

.. …

.. …

n. Viene decifrato il VALORE_CIFRATO 1 con la PUBLIC_KEY ottenuta al passo precedente ottenendo un certo valore. Se suddetto valore corrisponde all'ID_OGGETTO dichiarato sul TAG allora le informazioni inerenti il percorso compiuto dall'oggetto, dichiarate mediante la sequenza di ID_TAPPA presenti sul TAG, non sono state alterate

2.2.3 Cancellazione e Clonazione dei Dati sul TAG

La ricerca di soluzioni ai problemi relativi alla possibilità di cancellare e di alterare i dati sul TAG ha dato esito negativo viste le premesse con cui è stato ideato il sistema di tracciamento. Più in particolare, la cancellazione dei dati risulta essere un problema intrinsecamente legato all'utilizzo di TAG scrivibili, ipotesi fondamentale nell'architettura del sistema di tracciamento da noi ideato. Tuttavia, dato che il nostro scopo è quello di rilevare la manomissione delle informazioni relative al percorso di un oggetto, questo ci è sembrato un problema di secondaria importanza.

Un discorso differente merita invece quello riguardante la possibilità di clonare i TAG. Infatti, poiché lo scopo con cui abbiamo deciso di immagazzinare direttamente sul TAG le informazioni sul percorso dell'oggetto è fondamentalmente quello di dare la possibilità a chiunque abbia a disposizione un dispositivo in grado di effettuare la lettura del TAG di effettuare un controllo sull'origine dell'oggetto in questione, si rende necessario trovare una soluzione a suddetto problema. Una soluzione a suddetto problema può essere quella di impiegare come ID_OGGETTO il numero di serie di cui ogni TAG è dotato il cui valore, scritto nel momento della fabbricazione dello stesso, non può essere alterato tramite l'impiego di alcun tipo di dispositivo. Dopo quanto detto, non risulta più necessario generare un apposito ID_OGGETTO per identificare l'oggetto e, pertanto, sul TAG risulta necessario scrivere unicamente i valori indicati nella seguente tabella.

ID_TAPPA 1
...
ID_TAPPA N
VALORE_DI_CHECK

2.2.4 Considerazioni sulle Scelte Adoperate

Il VALORE_DI_CHECK, per come abbiamo scelto di determinarlo, in maniera simile ad una firma digitale

- permette di verificare che le informazioni presenti sul TAG non siano state alterate, dunque, che non siano stati alterati l'ID_OGGETTO e l'elenco dei campi del tipo ID_TAPPA
- permette di verificare che ciascun campo del tipo ID_TAPPA sia stato aggiunto proprio in corrispondenza dell'arrivo dell'oggetto presso la tappa a cui è associato quel determinato ID_TAPPA
- impedisce che si possa negare che un certo ID_TAPPA sia stato aggiunto proprio in corrispondenza dell'arrivo dell'oggetto presso la tappa a cui è associato quel determinato ID_TAPPA

ma oltre a ciò garantisce anche che la sequenza degli ID_TAPPA riportati sul TAG sia quella con cui è stato effettivamente aggiornato il VALORE_DI_CHECK, in altri termini che la sequenza delle tappe attraverso le quali l'oggetto è passato corrisponde esattamente a quella riportata sul TAG.

2.3 Casi d'Uso

In base a quanto visto finora risulta possibile riassumere le funzionalità del sistema nel seguente diagramma dei casi d'uso.

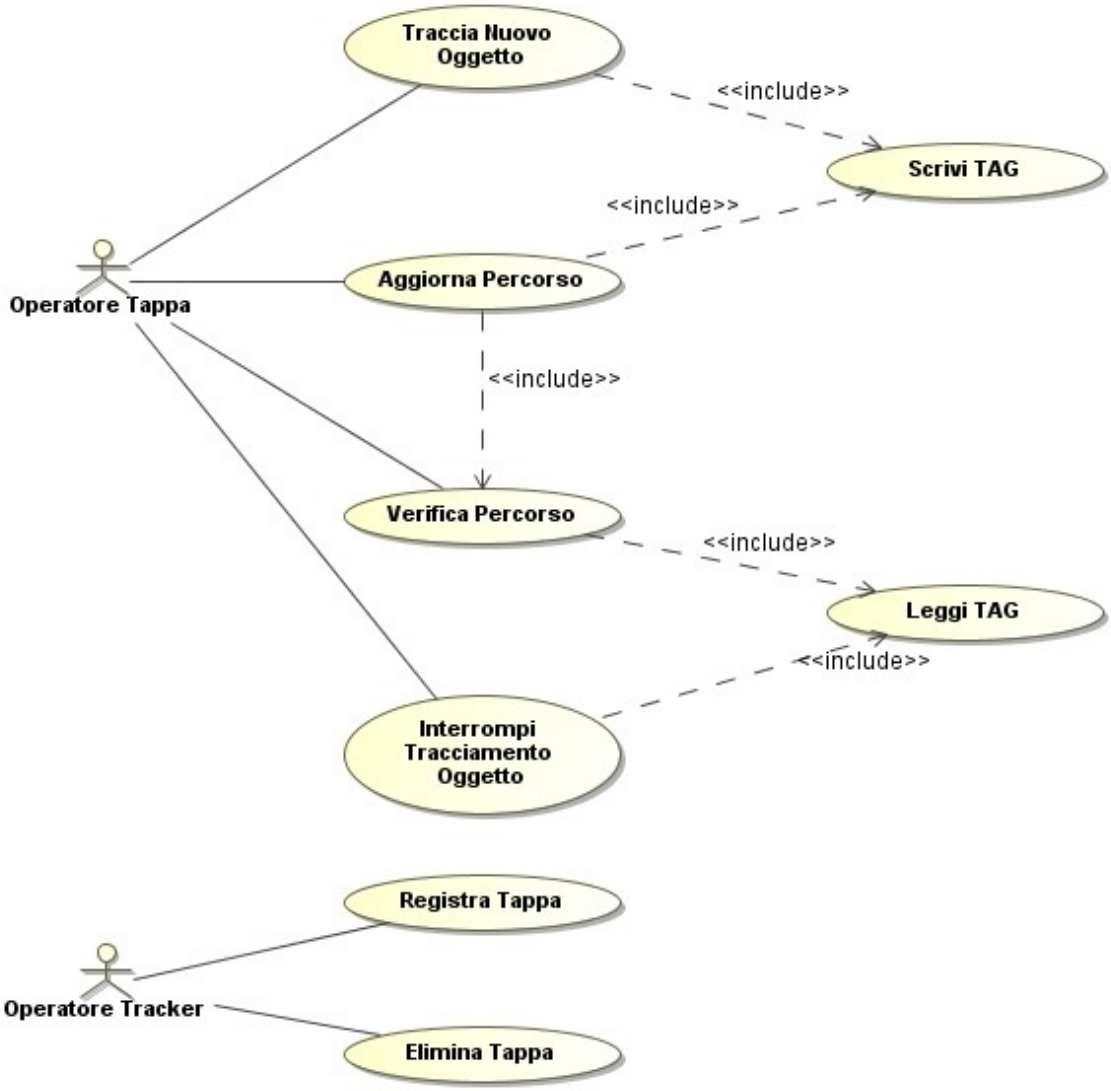


Diagramma dei Casi d'Uso

Ora, di seguito, la descrizione di ciascuno dei casi d'uso presenti nel precedente diagramma.

Caso d'uso: Registra Tappa

Scopo: Registrare presso il *Tracker* una certa tappa in modo che gli sia consentito l'aggiornamento del percorso compiuto dagli oggetti

Iniziatore: Operatore *Tracker*

Descrizione:
1. L'operatore richiede al *Tracker* di effettuare la registrazione di una nuova tappa
2. Il *Tracker* risponde richiedendo all'operatore di inserire la PUBLIC_KEY precedentemente fornitagli da un operatore della tappa
3. L'operatore fornisce al sistema la PUBLIC_KEY richiesta
4. Il *Tracker* genera un identificativo univoco per la tappa di cui è stata fatta richiesta di registrazione, l'ID_TAPPA, e lo associa alla PUBLIC_KEY precedentemente fornita creando un un nuovo record da inserire nella *Tabella Tappe*
5. Il *Tracker* conclude la procedura di registrazione restituendo all'operatore l'ID_TAPPA precedentemente creato. L'operatore del *Tracker* si occuperà di comunicare in un secondo momento ad un operatore della tappa appena registrata l'ID_TAPPA

Caso d'uso: Elimina Tappa

Scopo: Eliminare la registrazione di una determinata tappa dal *Tracker*

Iniziatore: Operatore *Tracker*

Descrizione:
1. L'operatore richiede al *Tracker* di iniziare la procedura di eliminazione della registrazione di una certa tappa
2. Il sistema risponde richiedendo l'inserimento dell'ID_TAPPA corrispondente alla tappa di cui eliminare la registrazione
3. L'operatore fornisce l'ID_TAPPA richiesto
4. Il sistema effettua una ricerca all'interno della *Tabella Tappe* utilizzando come chiave di ricerca l'ID_TAPPA fornitogli dall'operatore. Se la ricerca ha esito positivo allora il sistema procede con l'eliminazione del record, all'interno della *Tabella Tappe*, in cui compare l'ID_TAPPA fornito dall'operatore del *Tracker*
5. Il sistema invia una notifica all'operatore del *Tracker* circa il successo o l'insuccesso dell'operazione

Caso d'uso: Traccia Nuovo Oggetto

Scopo: Generare le informazioni per il tracciamento di un nuovo oggetto che andranno memorizzate sul TAG e presso il *Tracker*

Iniziatore: Operatore Tappa

Descrizione:
1. L'operatore della tappa acquisisce il numero di serie univoco del TAG tramite il dispositivo con cui è equipaggiato
2. L'operatore della tappa contatta il *Tracker* comunicandogli di iniziare il processo di tracciamento di un nuovo oggetto, identificato da un ID_OGGETTO con valore pari a quello del numero di serie del TAG
3. L'operatore della tappa, tramite un dispositivo con cui è equipaggiato, genera le informazioni da scrivere sul TAG. Suddette informazioni saranno costituite, più in particolare, dalla sequenza costituita dall'ID_TAPPA e dalla cifratura, mediante la chiave privata associata alla tappa, dell'ID_OGGETTO, ossia, quello che abbiamo chiamato VALORE_DI_CHECK

Caso d'uso: Verifica Percorso

Scopo: Verificare l'autenticità del percorso compiuto dall'oggetto in transito e dichiarato sul TAG

Iniziatore: Operatore Tappa

Descrizione:
1. L'operatore della tappa, tramite il dispositivo di cui è dotato, effettua la lettura delle informazioni presenti sul TAG (Caso d'Uso Leggi TAG)
2. L'operatore intraprende una ricerca delle PUBLIC_KEY associate agli ID_TAPPA dichiarati sul TAG presso il *Tracker*. Se non risulta possibile reperire anche una sola delle PUBLIC_KEY richieste allora si considerano compromesse le informazioni presenti sul TAG
3. L'operatore della tappa, mediante il dispositivo di cui è dotato e tramite le PUBLIC_KEY reperite presso il *Tracker*, veridica la validità del VALORE_DI_CHECK acquisito in precedenza dal TAG. Se suddetto controllo da esito negativo allora si considerano compromesse le informazioni presenti sul TAG

Caso d'uso: Aggiorna Percorso

Scopo: Aggiornare le informazioni circa il percorso compiuto dall'oggetto

Iniziatore: Operatore Tappa

Descrizione:
1. L'operatore della tappa verifica l'autenticità delle informazioni presenti sul TAG (Caso d'Uso Verifica Percorso)
2. Viene aggiornato, presso il *Tracker*, il percorso compiuto dall'oggetto in questione mediante l'aggiunta dell'ID_TAPPA della tappa in cui si trova al momento l'oggetto all'elenco degli ID_TAPPA associati all'ID_OGGETTO
3. Viene generato il nuovo VALORE_DI_CHECK, costituito dalla cifratura, mediante la chiave privata della tappa, del VALORE_DI_CHECK precedentemente riportato sul TAG
4. Si procede con l'aggiornamento dei valori sul TAG (Caso d'Uso Scrivi TAG)

Caso d'uso: Leggi TAG

Scopo: Acquisire le informazioni inerenti l'oggetto presenti sul TAG

Iniziatore: Operatore Tappa

Descrizione: Vengono acquisiti dal TAG l'elenco degli ID_TAPPA ed il VALORE_DI_CHECK per il controllo dell'autenticità del percorso dichiarato sul TAG

Caso d'uso: Scrivi TAG

Scopo: Inserire sul TAG le informazioni per l'identificazione dell'oggetto e quelle riguardanti il percorso da esso compiuto

Iniziatore: Operatore Tappa

Descrizione: Vengono scritti sul TAG l'elenco degli ID_TAPPA ed il VALORE_DI_CHECK

Caso d'uso: Interrompi Tracciamento Oggetto

Scopo: Interrompere il processo di tracciamento di un determinato oggetto

Iniziatore: Operatore Tappa

Descrizione: Viene informato il *Tracker* della volontà di voler interrompere il tracciamento dell'oggetto in questione. A seconda dei casi si potrà inoltre scegliere se eliminare anche le informazioni presenti sul TAG oppure lasciarle inalterate per consentire anche successivamente di verificare la provenienza dell'oggetto

2.4 Diagrammi delle Classi

Come abbiamo già avuto modo di capire da quanto detto fin ora, un sistema di tracciamento basato su tecnologia RFID si presenterà come un sistema strutturato fondamentalmente in due parti:

- Un *Unità di Tracciamento*, altrimenti denominata *Tracker*, il cui compito è quello di tener traccia della posizione e dello stato degli oggetti
- Una componente Client, in grado di interfacciarsi sia con il lettore RFID che con l'*Unità di Tracciamento*

Nel corso di questo capitolo ci siamo proposti di delineare l'architettura di un sistema di tracciamento utile unicamente al monitoraggio della posizione di un oggetto senza cioè preoccuparci di tener traccia di eventuali informazioni riguardanti l'evoluzione dello stato dell'oggetto durante i suoi spostamenti. Di seguito riportiamo i diagrammi delle classi in UML di quelle che sono le due componenti del sistema di cui abbiamo poco fa fatto parola nei quali non si troverà traccia, quindi, di strutture utili al tracciamento dello stato dell'oggetto.

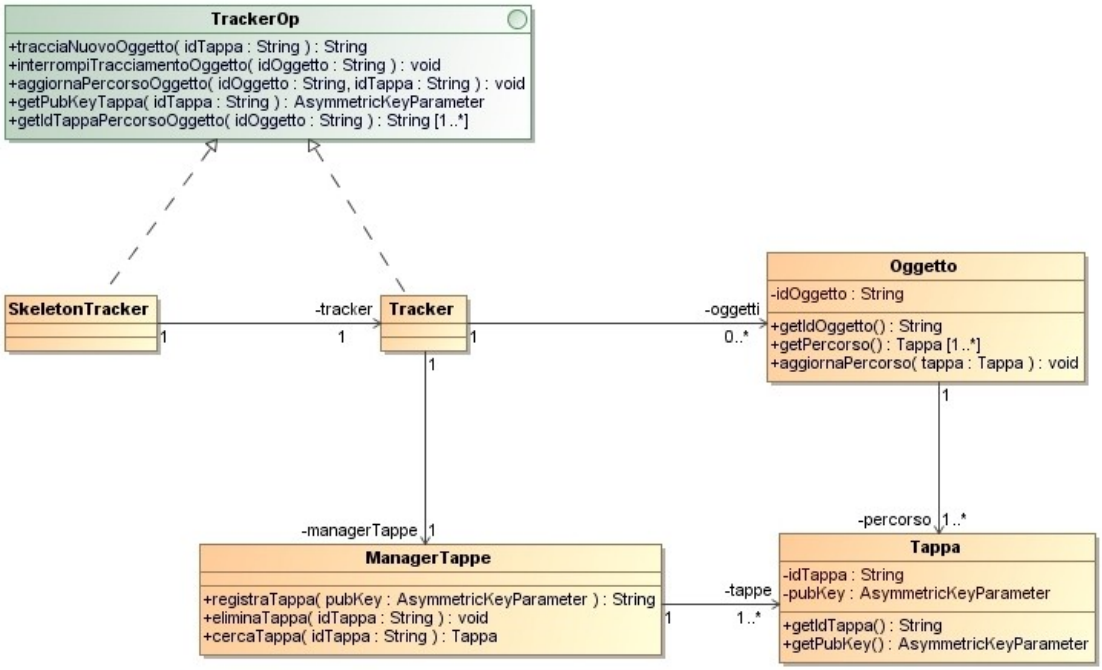


Diagramma delle Classi dell'Unità di Tracciamento

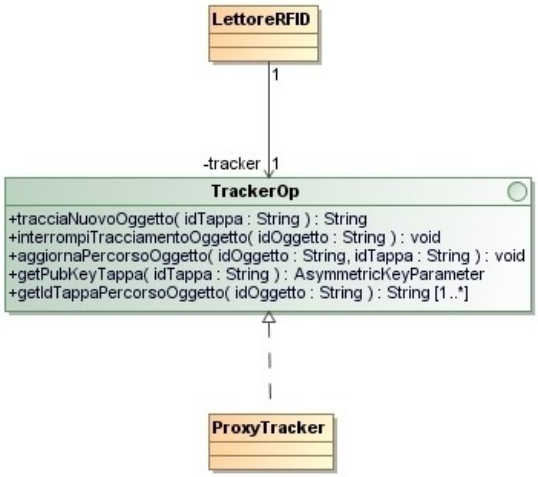
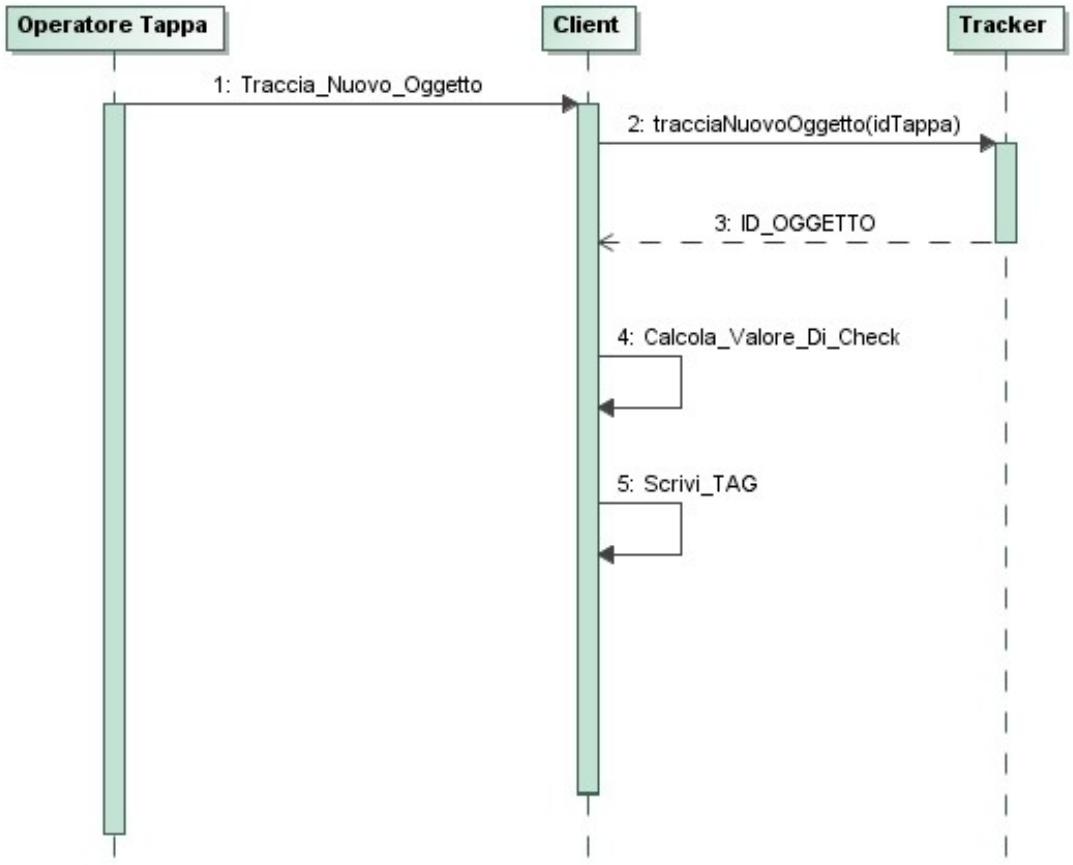


Diagramma delle Classi del Client

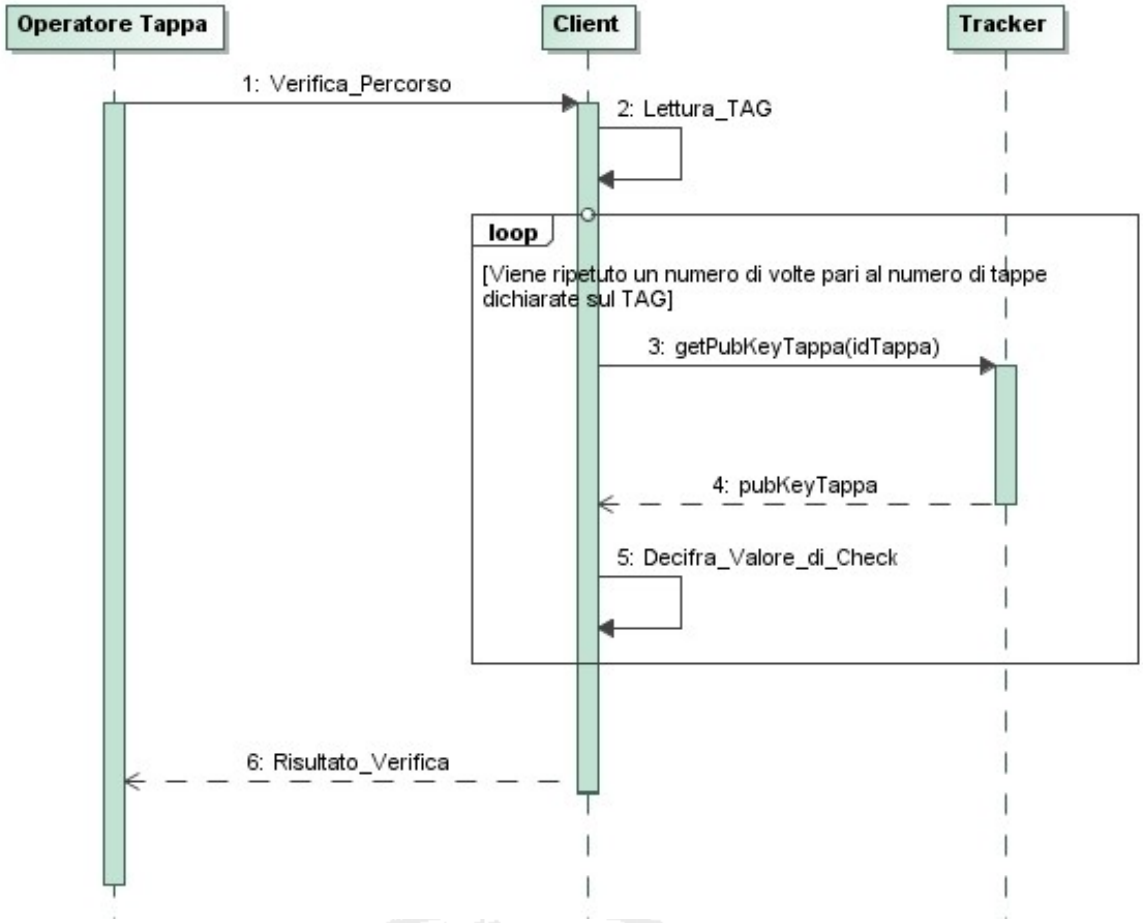
2.5 Scenari d'Utilizzo

Ora che abbiamo delineato tutte le componenti del nostro sistema di tracciamento siamo dunque in grado di mostrarne alcuni scenari d'utilizzo. Cominceremo illustrando in che modo le varie entità del nostro sistema si scambiano messaggi al fine di dare inizio al processo di tracciamento di un nuovo oggetto. Questo è illustrato di seguito attraverso un diagramma di sequenza.

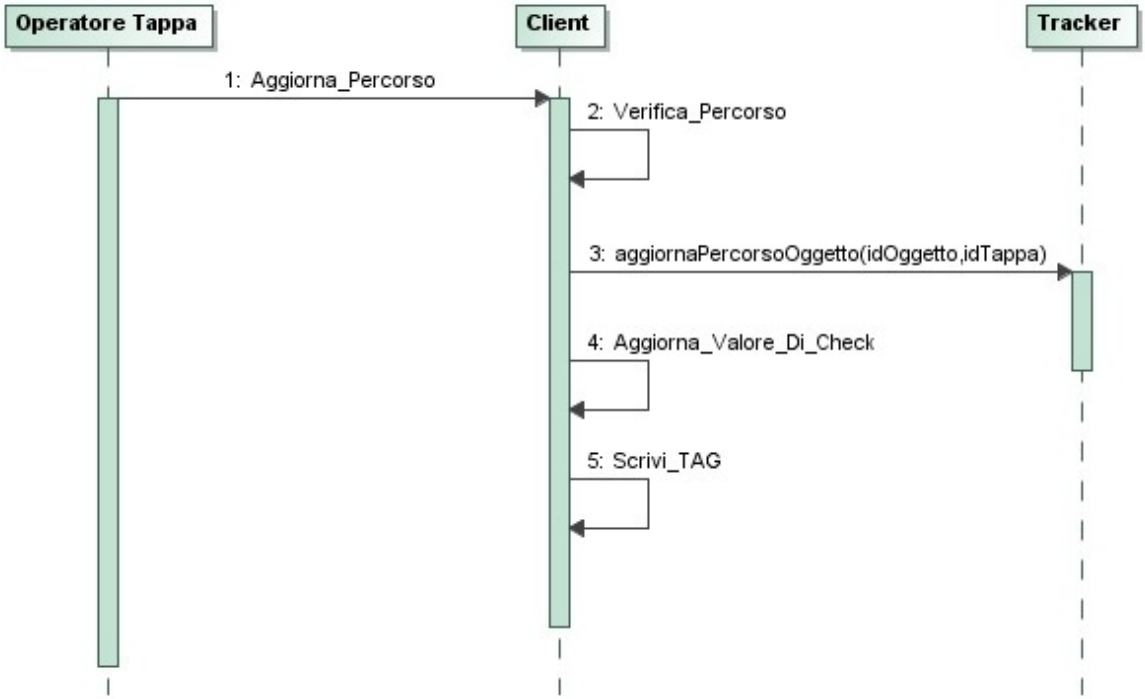


Illustriamo ora l'interazione tra le entità del sistema durante il controllo dell'autenticità del percorso compiuto da un oggetto tracciato. Suddetto controllo si articolerà innanzitutto nel controllo della coerenza dei valori presenti sul TAG sfruttando quello che abbiamo chiamato VALORE_DI_CHECK. Successivamente, nel caso in cui la precedente operazione si concluda con esito positivo, si provvederà a confrontare il percorso dichiarato sul TAG con quello dichiarato invece dal *Tracker* con lo scopo di assicurarci

che non ci siano state manomissioni su quest'ultimo.



Una volta che è stato dato inizio al processo di tracciamento di un oggetto si rende necessario aggiornare le informazioni relative alla sua posizione ogniqualvolta giunge in una nuova tappa. Il processo di aggiornamento della posizione di un oggetto sia presso il *Tracker* che sul TAG attaccato all'oggetto stesso è illustrato invece nel seguente diagramma di sequenza.


Operatore Tappa
Client
Tracker
1: Aggiorna_Percorso
2: Verifica_Percorso
3: aggiornaPercorsoOggetto(idOggetto,idTappa)
4: Aggiorna_Valore_Di_Check
5: Scrivi_TAG

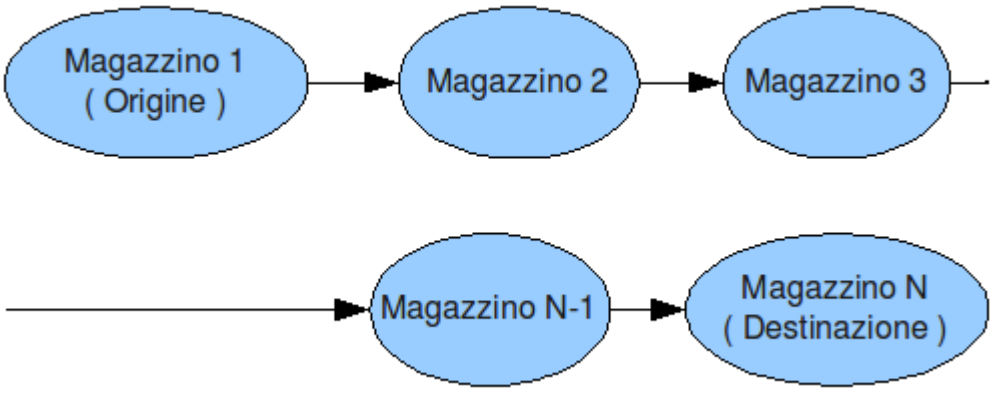
Capitolo 3

Un Sistema per il Tracciamento Sicuro delle Merci

Nel precedente capitolo abbiamo delineato quelle che sono le linee guida per lo sviluppo di un sistema di tracciamento oggetti basato su tecnologia RFID strizzando in particolar modo l'occhio ai problemi di sicurezza che si pongono nello sviluppo di un simile sistema. Ci proponiamo, invece, in questo capitolo di illustrare il modo in cui i concetti esposti fino ad ora possono essere messi in pratica per l'implementazione di un prototipo di applicazione per il tracciamento merci.

3.1 Cosa Vogliamo Realizzare

Si vuole tracciare una certa merce lungo un tragitto che la porta dal suo luogo d'origine fino alla destinazione. Il tragitto che la merce compie si articola, in particolare, in una serie di magazzini secondo lo schema riportato di seguito il quale non rappresenta altro che una specializzazione di uno schema equivalente già illustrato nel capitolo precedente.



Il percorso di una Spedizione in un Sistema di Tracciamento basato su RFID

Le informazioni inerenti il tragitto compiuto dalla merce fino ad un certo istante e quindi le informazioni sui diversi magazzini attraverso i quali la merce è passata sono inserite sia in un TAG applicato alla merce sia su quella che abbiamo chiamato fino ad ora *Unità di*

Tracciamento. Ogni volta che la merce giunge in un magazzino uno dei magazzinieri aggiorna, tramite un apposito dispositivo con il quale è equipaggiato, con le informazioni circa il magazzino nella quale la merce si trova sia il TAG che l'*Unità di Tracciamento*. La suddetta operazione deve però avvenire previo controllo dell'autenticità delle informazioni già presenti nel TAG. Più in particolare deve essere effettuato un controllo del percorso compiuto dalla merce prima di giungere nel magazzino corrente verificando che le informazioni inerenti i magazzini attraverso i quali la merce è passata in precedenza non siano state contraffatte.

3.2 Tecnologie Adottate e Strumenti di Sviluppo

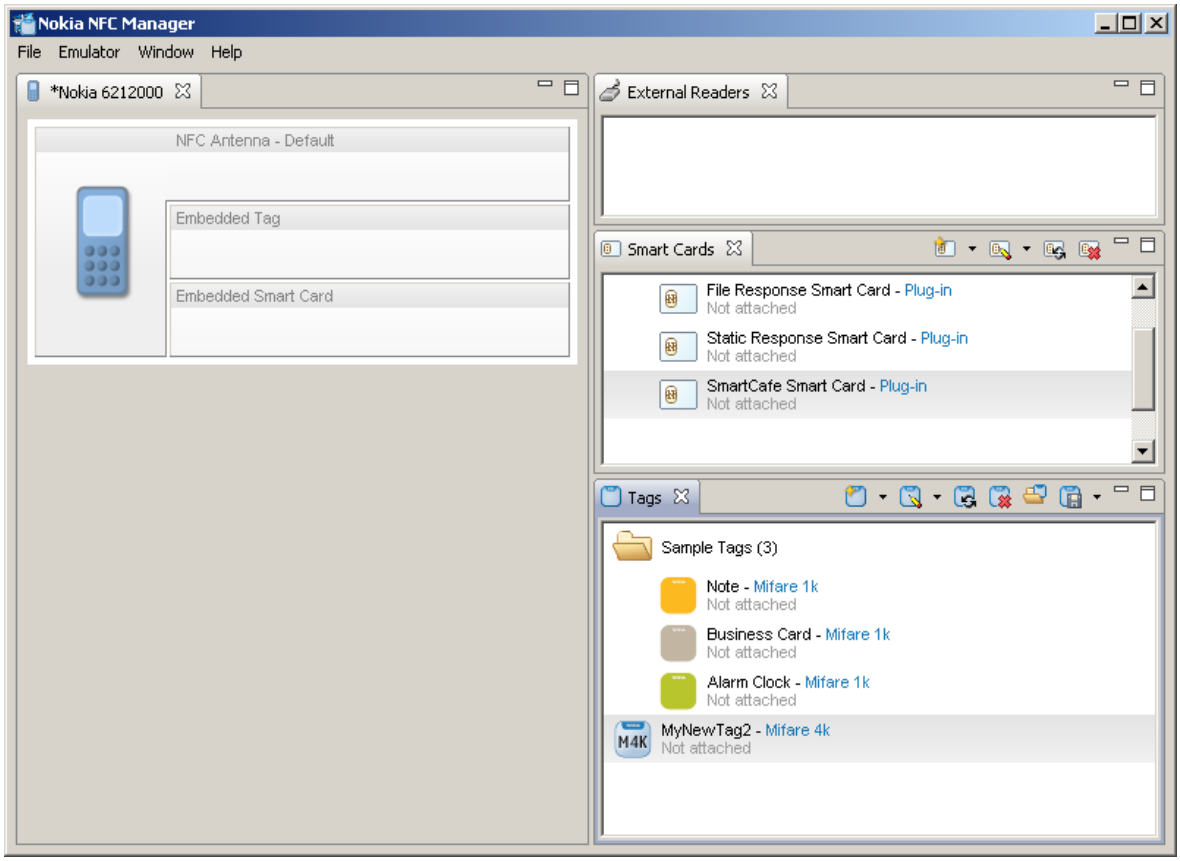
Dovendo memorizzare su TAG sia le informazioni per l'identificazione che quelle per il tracciamento del percorso di un oggetto si rende necessario innanzitutto un dispositivo che sia in grado di effettuare la lettura e la scrittura dei TAG RFID. Purtroppo non avendo a disposizione strumenti di questo genere abbiamo deciso di utilizzare come dispositivo lettore un telefono cellulare con integrata la tecnologia NFC la quale, come per l'RFID, ci permette di realizzare operazioni di identificazione in radiofrequenza degli oggetti a cui siamo interessati tramite la lettura dei TAG ad essi attaccati e per i cui dettagli si rimanda in appendice. Altro requisito indispensabile per lo sviluppo del nostro prototipo di applicazione di tracciamento merci è che ci venga fornita la possibilità di scrivere un applicazione in grado di essere eseguita da un siffatto dispositivo, tramite la quale andremo ad interagire con le funzionalità offerte dalla tecnologia NFC ed andremo ad eseguire tutte le altre operazioni che l'operatore del magazzino necessita di compiere. La nostra scelta è ricaduta quindi sul telefono cellulare Nokia 6212 classic il quale ci mette a disposizione

- tecnologia NFC e più in particolare la possibilità di interagire con TAG di tipo NFC Forum Type 1, NFC Forum Type 2, NFC Forum Type 3, NFC Forum Type 4, Innovision Jewel, Mifare 1K e 4K
- un ambiente Java ME per lo sviluppo e l'esecuzione di applicazioni basato su MIDP 2.1 (JSR 118) e CLDC 1.1 (JSR 139)

– le Contactless Communication API per Java (JSR 257) le quali consentono alle applicazioni Java in esecuzione sul dispositivo di interagire con Codici a Barre e TAG RFID

Non avendo fisicamente a disposizione il dispositivo in questione ci siamo avvalsi dell'emulatore integrato nel Nokia 6212 NFC SDK, il kit di sviluppo che la Nokia mette a disposizione per lo sviluppo ed il testing delle applicazioni sul Nokia 6212 classic.

L'emulatore del Nokia 6212 classic



Il Nokia NFC Manager

Abbiamo parlato, nel precedente capitolo, della necessità di adottare un modo per verificare l'eventuale manomissione delle informazioni sul TAG ed abbiamo visto come gli strumenti che la crittografia moderna ci mette a disposizione rappresentano un valido alleato. Ci siamo quindi serviti di questo tipo di strumenti nell'implementazione dell'applicazione da integrare sul dispositivo mobile da noi scelto. Più in particolare per la realizzazione delle operazioni crittografiche sul dispositivo è stata effettuata tramite le API crittografiche Bouncy Castle per i cui dettagli si rimanda in appendice.

Per quanto riguarda lo sviluppo dell'*Unità di Tracciamento* ci siamo invece avvalsi della piattaforma Java SE la quale mette a disposizione un insieme di API che raccolgono una gran quantità di componenti disponibili per scrivere applicazioni di qualsiasi genere oltre a fornire quello che rappresenta forse il più famigerato dei vantaggi offerti da questa piattaforma di sviluppo ossia la possibilità di poter eseguire il codice compilato, il cosiddetto bytecode, su qualsiasi piattaforma hardware sulla quale sia disponibile un implementazione della Java Virtual Machine.

3.3 Realizzazione del Sistema

Daremo ora uno sguardo a quelli che sono i dettagli relativi all'implementazione del sistema che ci siamo proposti di realizzare e soprattutto alla maniera con cui sono stati ripresi i concetti relativi all'architettura di un generico sistema di tracciamento esposti nel precedente capitolo per la realizzazione dello scopo che qui ora ci proponiamo di raggiungere.

3.3.1 I Dati Necessari al Tracciamento

I concetti esposti nel precedente capitolo possono considerarsi tutti validi ed applicabili al caso di nostro interesse. Le uniche variazioni che qui andremo a proporre prevedono unicamente cambiamenti nella terminologia utilizzata. Denoteremo pertanto gli identificatori tramite i quali verranno identificati rispettivamente la merce in spedizione e il magazzino attraverso il quale la merce transita con ID_SPEDIZIONE, il cui valore sarà pari al numero di serie del TAG, e ID_MAGAZZINO. L'*Unità di Tracciamento*, che nel contesto in cui operiamo sarebbe più opportuno chiamare *Tracker Spedizioni*, gestisce le informazioni riguardanti le spedizioni tracciate e l'elenco dei magazzini attraverso due tabelle: la *Tabella Spedizioni* e la *Tabella Magazzini*.

Tabella Spedizioni	
ID_SPEDIZIONE 1	ID_MAGAZZINO 1 ID_MAGAZZINO 2 … ID_MAGAZZINO N
…	…
ID_SPEDIZIONE M	...

Tabella Magazzini	
ID_MAGAZZINO 1	Informazioni per l'identificazione dello specifico magazzino
...	...
ID_MAGAZZINO N	...

Da quello che abbiamo fin ora detto risulta facile intuire che l'insieme dei valori che andranno scritti sul TAG da apporre alla merce in transito sarà quello presente nella seguente tabella nella quale il campo VALORE_DI_CHECK sarà determinato mediante lo stesso algoritmo illustrato nel capitolo precedente.

ID_MAGAZZINO 1
...
ID_MAGAZZINO N
VALORE_DI_CHECK

3.3.2 Diagramma dei Casi d'Uso

Le funzionalità che il sistema dovrà in definitiva esporre possono essere illustrate schematicamente dal seguente diagramma dei casi d'uso.

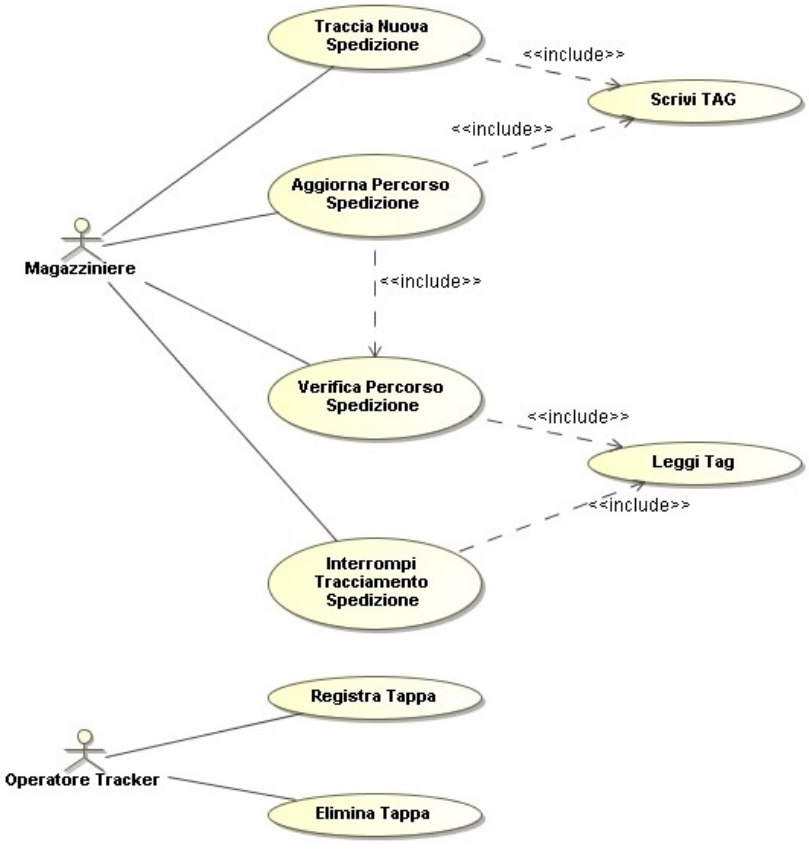


Diagramma dei Casi d'Uso

3.3.3 Diagrammi delle Classi

Presentiamo, a questo punto, i diagrammi delle classi rispettivamente del *Tracker Spedizioni* e dell'applicazione che andremo ad installare sul cellulare Nokia 6212 classic. Un occhio attento noterà senz'altro che i seguenti diagrammi non rappresentano altro che una specializzazione di quelli già presentati nel capitolo precedente.

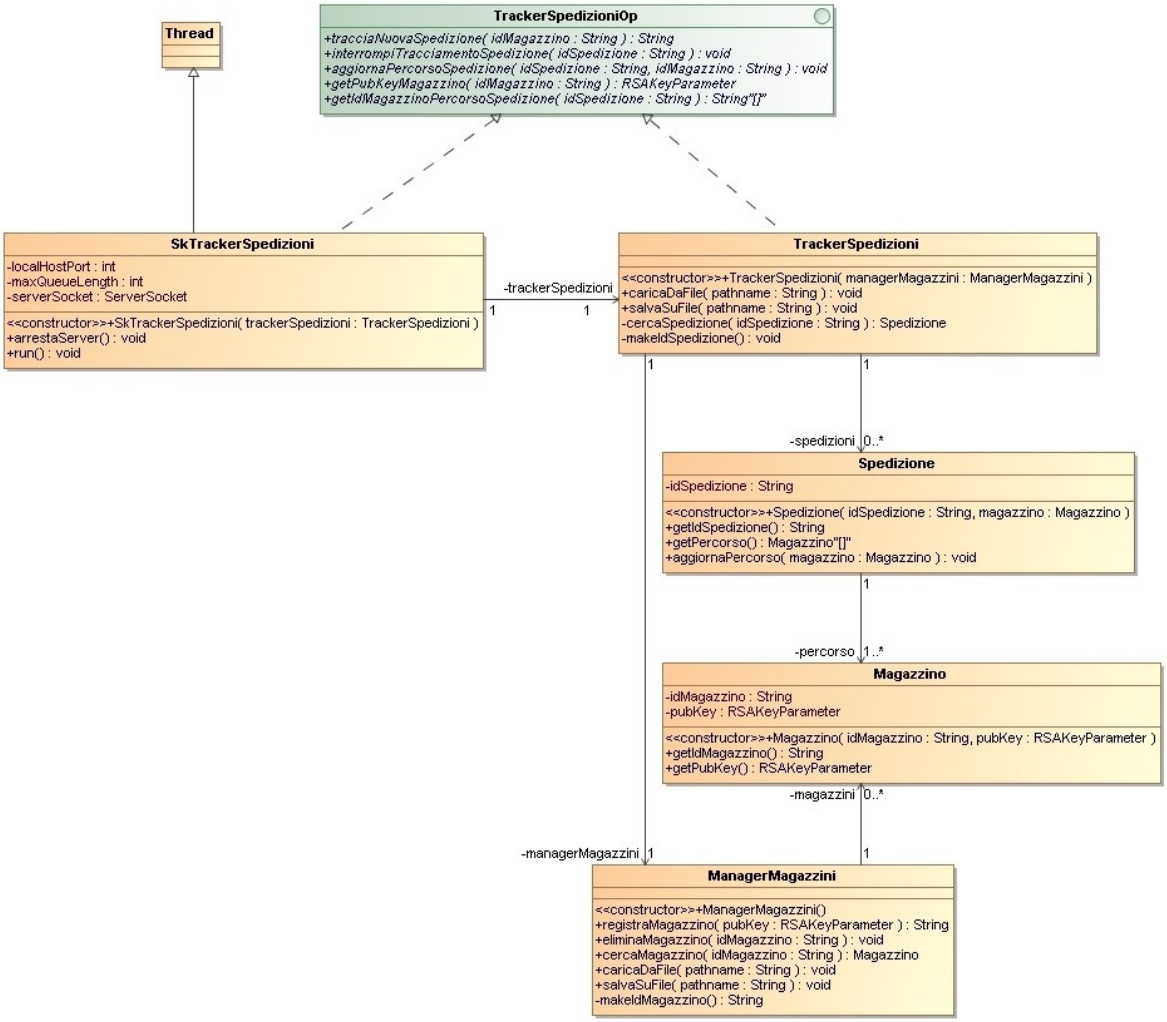


Diagramma delle Classi del Tracker Spedizioni

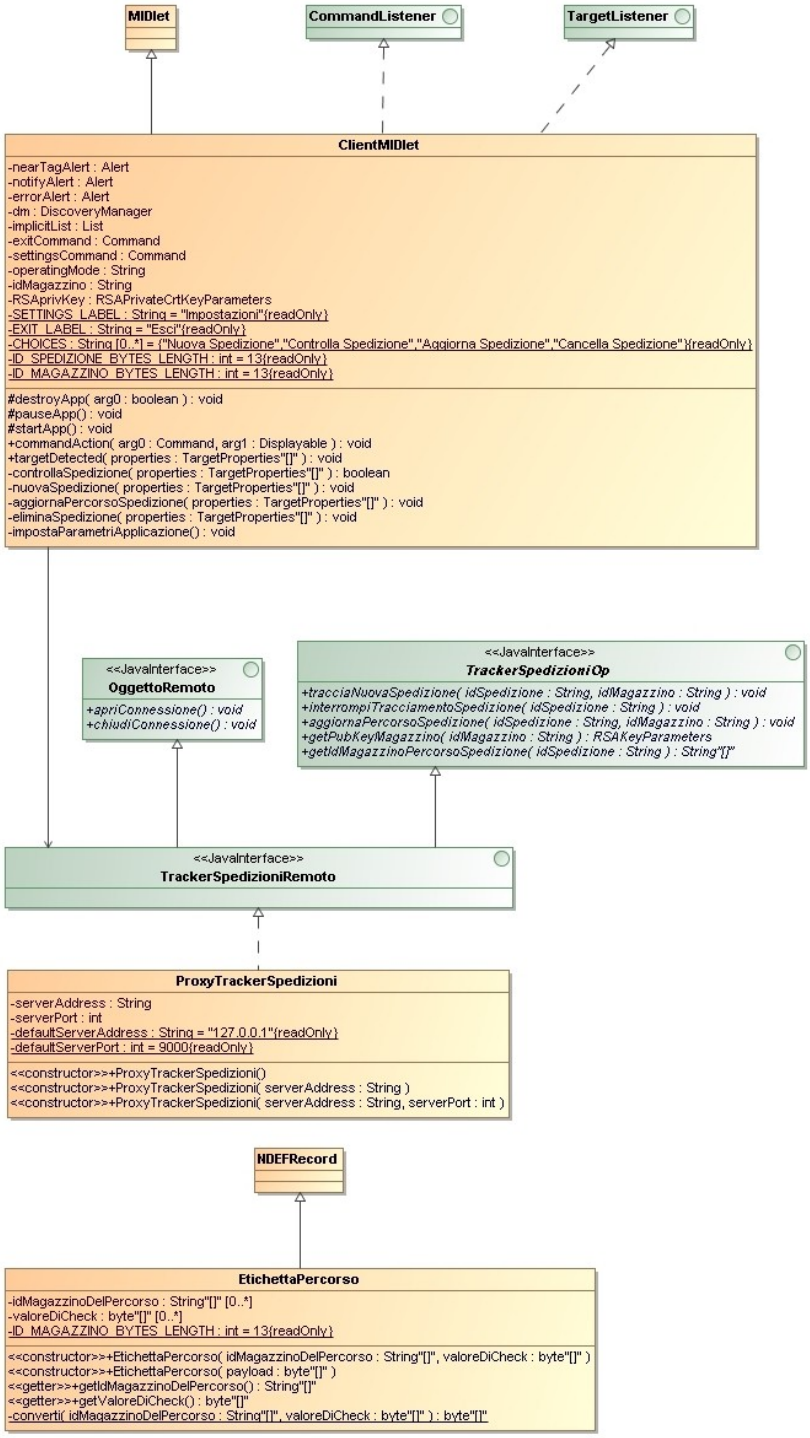
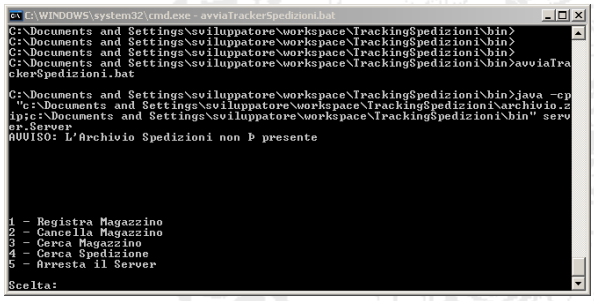


Diagramma delle Classi del Client

3.4 Scenari d'Utilizzo

Concluderemo la descrizione del nostro sistema di tracciamento mostrando, tramite l'ausilio di alcuni screenshot, in che modo il sistema per la rilevazione delle alterazioni

delle informazioni sul TAG opera nella pratica. Prima di fare ciò però mostriamo in che modo l'applicazione, una volta installata sul Nokia 6212, permette di interagire con il TAG e con il *Tracker Spedizioni* per svolgere le operazioni di tracciamento di una spedizione. Iniziamo col mostrare in che modo avviene l'inizio del tracciamento di una certa merce che ipotizziamo si trovi inizialmente in un certo magazzino per il quale è stata precedentemente effettuata la procedura di registrazione presso il *Tracker Spedizioni*. L'applicazione sul dispositivo mobile è stata opportunamente preimpostata con i parametri necessari al suo funzionamento: l'ID_MAGAZZINO del magazzino nel quale si ipotizza si stia operando e la chiave privata associata a suddetto magazzino.

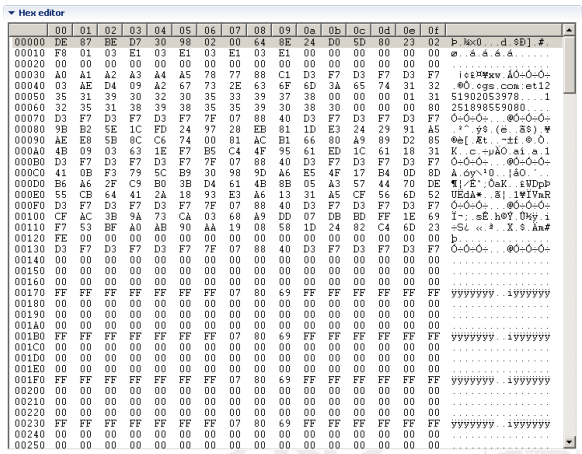


Come possiamo vedere il *Tracker Spedizioni* è in esecuzione

Anche l'applicazione sul nostro Nokia 6212 è in esecuzione

Quando viene selezionata la voce *Nuova Spedizione* viene richiesto all'utente di avvicinare il dispositivo al TAG

Il dispositivo notifica all'utente che la spedizione è stata creata con successo



Dopo la creazione della nuova spedizione è possibile notare le informazioni che sono state scritte sul TAG

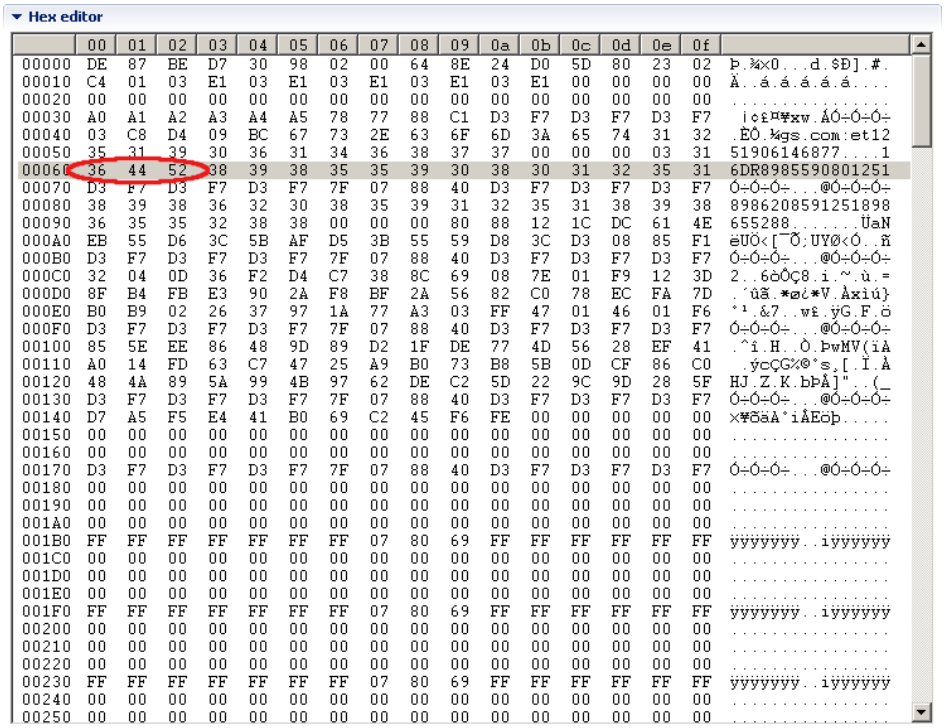
Le altre operazioni che l'applicazione rende disponibili possono essere compiute con la stessa facilità con cui è stata effettuata la creazione di una nuova spedizione. Vediamo ora in che modo si comporta il nostro dispositivo quando viene messa in atto una contraffazione sul TAG.

Hex editor

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000	DE	87	BE	D7	30	98	02	00	64	8E	24	D0	5D	80	23	02	Þ.¾×0...d.$Ð].#.
00010	C4	01	03	E1	03	E1	03	E1	03	E1	03	E1	00	00	00	00	Ä..á.á.á.á.á....
00020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00030	A0	A1	A2	A3	A4	A5	78	77	88	C1	D3	F7	D3	F7	D3	F7	¡¢£¤¥xw.ÁÓ÷Ó÷Ó÷
00040	03	C8	D4	09	BC	67	73	2E	63	6F	6D	3A	65	74	31	32	.ÈÔ.¼gs.com:et12
00050	35	31	39	30	36	31	34	36	38	37	37	00	00	00	03	31	51906146877....1
00060	32	35	31	38	39	38	35	35	39	30	38	30	31	32	35	31	2518985590801251
00070	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00080	38	39	38	36	32	30	38	35	39	31	32	35	31	38	39	38	89862085912518 98
00090	36	35	35	32	38	38	00	00	00	80	88	12	1C	DC	61	4E	655288.....ÜaN
000A0	EB	55	D6	3C	5B	AF	D5	3B	55	59	D8	3C	D3	08	85	F1	ëUÖ<[¯Õ;UYØ<Ó..ñ
000B0	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
000C0	32	04	0D	36	F2	D4	C7	38	8C	69	08	7E	01	F9	12	3D	2..6òÔÇ8.i.~.ù.=
000D0	8F	B4	FB	E3	90	2A	F8	BF	2A	56	82	C0	78	EC	FA	7D	.´ûã.*ø¿*V.Àxìú}
000E0	B0	B9	02	26	37	97	1A	77	A3	03	FF	47	01	46	01	F6	°¹.&7.w£.ÿG.F.ö
000F0	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00100	85	5E	EE	86	48	9D	89	D2	1F	DE	77	4D	56	28	EF	41	.^î.H..Ò.ÞwMV(iA
00110	A0	14	FD	63	C7	47	25	A9	B0	73	B8	5B	0D	CF	86	C0	.ýcÇG%©°s¸[.Ï.À
00120	48	4A	89	5A	99	4B	97	62	DE	C2	5D	22	9C	9D	28	5F	HJ.Z.K.bÞÂ]".(_
00130	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00140	D7	A5	F5	E4	41	B0	69	C2	45	F6	FE	00	00	00	00	00	×¥õäA°iÂEöþ.....
00150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00170	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001B0	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001F0	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
00200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00230	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
00240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Possiamo notare come sul TAG sono presenti tutte le informazioni riguardanti il tragitto della merce al quale è attaccato

Il controllo va a buon fine quindi la spedizione non sembra essere stata contraffatta



	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000	DE	87	BE	D7	30	98	02	00	64	8E	24	D0	5D	80	23	02	Þ.¾×0...d.$Ð].#.
00010	C4	01	03	E1	03	E1	03	E1	03	E1	03	E1	00	00	00	00	Ä..á.á.á.á.á....
00020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00030	A0	A1	A2	A3	A4	A5	78	77	88	C1	D3	F7	D3	F7	D3	F7	¡¢£¤¥xw.ÁÓ÷Ó÷Ó÷
00040	03	C8	D4	09	BC	67	73	2E	63	6F	6D	3A	65	74	31	32	.ÈÔ.¼gs.com:et12
00050	35	31	39	30	36	31	34	36	38	37	37	00	00	00	03	31	51906146877....1
00060	36	44	52	38	39	38	35	35	39	30	38	30	31	32	35	31	6DR89855908012 51
00070	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00080	38	39	38	36	32	30	38	35	39	31	32	35	31	38	39	38	89862085912518 98
00090	36	35	35	32	38	38	00	00	00	80	88	12	1C	DC	61	4E	655288.....ÜaN
000A0	EB	55	D6	3C	5B	AF	D5	3B	55	59	D8	3C	D3	08	85	F1	ëUÖ<[¯Õ;UYØ<Ó..ñ
000B0	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
000C0	32	04	0D	36	F2	D4	C7	38	8C	69	08	7E	01	F9	12	3D	2..6òÔÇ8.i.~.ù.=
000D0	8F	B4	FB	E3	90	2A	F8	BF	2A	56	82	C0	78	EC	FA	7D	.´ûã.*ø¿*V.Àxìú}
000E0	B0	B9	02	26	37	97	1A	77	A3	03	FF	47	01	46	01	F6	°¹.&7..w£.ÿG.F.ö
000F0	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00100	85	5E	EE	86	48	9D	89	D2	1F	DE	77	4D	56	28	EF	41	.^î.H..Ò.ÞwMV(iA
00110	A0	14	FD	63	C7	47	25	A9	B0	73	B8	5B	0D	CF	86	C0	.ýcÇG%©°s¸[.Ï.À
00120	48	4A	89	5A	99	4B	97	62	DE	C2	5D	22	9C	9D	28	5F	HJ.Z.K.bÞÂ]"..(_
00130	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00140	D7	A5	F5	E4	41	B0	69	C2	45	F6	FE	00	00	00	00	00	×¥õäA°iÂEöþ.....
00150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00170	D3	F7	D3	F7	D3	F7	7F	07	88	40	D3	F7	D3	F7	D3	F7	Ó÷Ó÷Ó÷...@Ó÷Ó÷Ó÷
00180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001B0	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
001F0	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
00200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00230	FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿ..iÿÿÿÿÿÿ
00240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Operiamo un alterazione dei byte evidenziati in figura

Possiamo vedere come l'alterazione precedentemente operata viene rilevata non appena viene effettuato un controllo del TAG

Conclusioni e Sviluppi Futuri

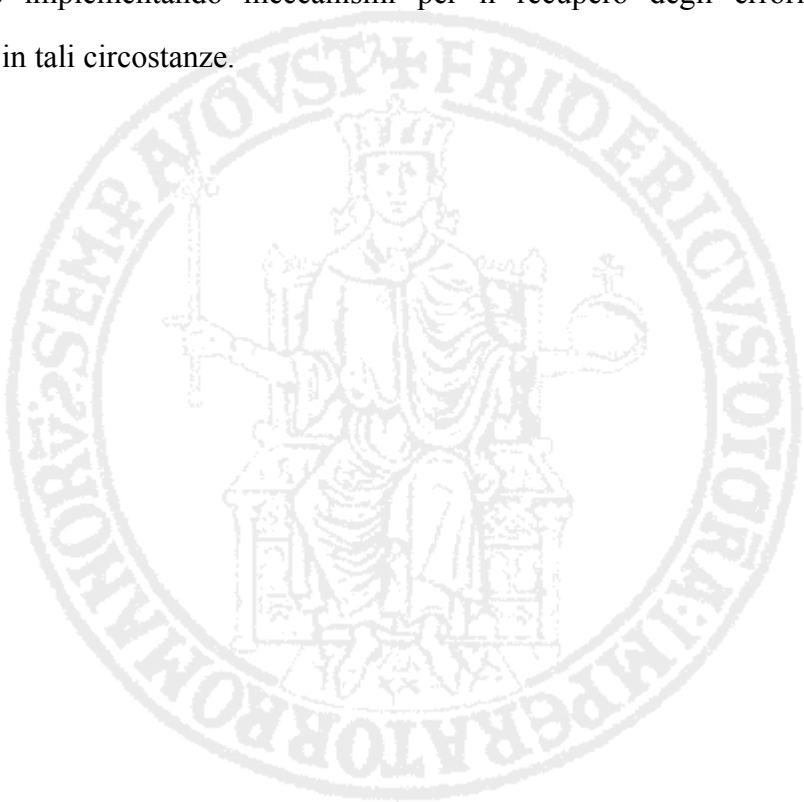
Il sistema da noi concepito inizialmente con lo scopo di permettere un tracciamento oggetti basato esclusivamente sull'utilizzo della tecnologia RFID non ha purtroppo dato i risultati sperati. Infatti, i problemi di sicurezza, ed in particolare il TAG Tampering, che si presentano nell'impiego di un simile approccio al tracciamento ci hanno impedito di ideare un architettura nella quale si potesse fare a meno di accedere all'entità *Tracker* per il recupero di informazioni, rappresentate nel nostro caso dalle chiavi pubbliche associate alle tappe del percorso. L'impiego di meccanismi di caching delle chiavi pubbliche dal lato di chi effettua il controllo del TAG risulterebbe, quindi, molto utile per limitate il numero di accessi al *Tracker*.

L'impiego di un simile sistema nell'ambito di un processo di tracciamento di filiera è altresì da escludersi. Il verificarsi di mancate letture già con l'impiego dei tipici sistemi di tracciamento con RFID, quelli nei quali viene effettuata la semplice lettura del numero di serie del TAG, ci fa infatti pensare a quello che potrebbe accadere mettendo in atto una comunicazione complessa come quella che avviene nel sistema da noi ideato.

L'unico vantaggio che ci viene in mente di citare è rappresentato dalla possibilità che chiunque sia dotato di un dispositivo in grado di leggere il TAG ha di controllare direttamente le informazioni riguardanti le tappe del percorso dell'oggetto esaminato senza che queste ultime vengano fornite da un terzo estraneo, il *Tracker*. Grazie infatti alla tecnica con cui viene determinato il VALORE_DI_CHECK chiunque esegue un controllo sul TAG può essere certo che le informazioni riguardanti ciascuna delle tappe dichiarate è stata inserita proprio in corrispondenza del passaggio dell'oggetto presso quella determinata tappa e che l'ordine con le quali le tappe sono indicate sul TAG corrisponde esattamente a quello con cui si è spostato l'oggetto attraverso il suo percorso. Queste osservazioni restano, tuttavia, valide unicamente nell'ipotesi in cui sia possibile considerare il *Tracker* un entità fidata, cioè in grado di fornire le autentiche chiavi pubbliche associate alle tappe identificate dalla sequenza degli ID_TAPPA dichiarata sul

TAG. Un idea che porterebbe alla risoluzione simultanea del problema della mancanza di affidabilità del *Tracker*, per quanto riguarda l'autenticità delle chiavi pubbliche distribuite e del problema delle perdite nello scambio dati tra TAG e Reader sarebbe quella di spostare sul *Tracker* l'elenco degli ID_TAPPA ed il VALORE_DI_CHECK e di impiegare in sostituzione delle semplici chiavi pubbliche dei certificati digitali emessi da una vera e propria autorità di certificazione.

Un ultima nota riguarda invece l'implementazione del sistema di tracciamento merci che abbiamo messo in piedi per mettere in pratica i concetti di cui tratta questa tesi. In suddetta implementazione non ci è stato possibile lavorare sugli aspetti riguardanti la sua robustezza ed, in particolar modo, sui meccanismi che regolano lo scambio dati tra l'applicazione che gira sul dispositivo mobile ed il *Tracker*. Non sono stati, infatti, eseguiti test sufficientemente approfonditi su come il sistema potrebbe reagire ad interruzioni nella comunicazione tra le suddette due componenti ed in particolare sulle incoerenze che si potrebbero venire a creare in queste circostanze nei dati riguardanti il percorso degli oggetti memorizzati sul TAG e sul *Tracker*. Si potrebbe, quindi, continuare a lavorare implementando meccanismi per il recupero degli errori che potrebbero verificarsi in tali circostanze.



Appendice A

Tecnologia NFC

L'NFC rappresenta una tecnologia per la connettività a corto raggio derivante dalla combinazione delle carte elettroniche senza contatto con altre tecnologie RFID. La promozione della standardizzazione e dell'implementazione dell'NFC avviene ad opera di un associazione di gruppi industriali, l'NFC Forum, di cui fanno ad oggi parte Sony, Philips, Samsung, Nokia, Visa, Mastercard e molti altri. La caratteristica distintiva di questa tecnologia è rappresentata dalle modalità con cui avviene la comunicazione tra dispositivi NFC compatibili. Al contrario di quanto avviene con i sistemi RFID, nei quali i TAG si limitano semplicemente a rispondere alle richieste del Reader, le entità coinvolte in una comunicazione NFC possono assumere anche pari dignità. Infatti un dispositivo NFC compatibile può funzionare secondo diverse modalità operative:

- Modalità Reader/Writer nella quale un dispositivo NFC si comporta esattamente come un Reader RFID con capacità quindi di leggere e scrivere in formazioni nei TAG NFC compatibili
- Modalità Card Emulation nella quale invece un dispositivo NFC emula il comportamento di una Smart Card Contactless di tipo Proximity Card secondo l'ISO/IEC 14443. In questa modalità di funzionamento un Reader non è in grado di distinguere il dispositivo NFC da una ordinaria Proximity Card
- Modalità Peer to Peer standardizzata nell'ISO/IEC 18092 che consente lo scambio dati tra due dispositivi NFC in una forma bidirezionale. Questa modalità operativa si differenzia quindi dalle precedenti due in quanto consente a due dispositivi NFC di avere pari dignità nella comunicazione

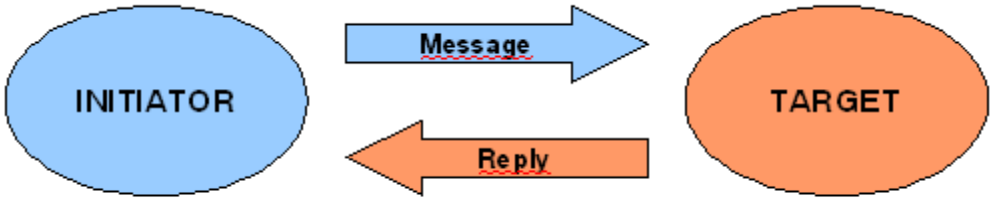
Oltre alle modalità con cui dispositivi NFC possono comunicare tra di loro la tecnologia NFC si presenta caratterizzata da

- una banda intorno ai 13,56 Mhz

- una distanza operativa di, al limite, 10cm
- velocità di trasmissione dati che può essere di 106Kbps, 212Kbps oppure 424Kbps

A.1 Comunicazione tra dispositivi NFC

Come già accennato in precedenza obbiettivo dell'NFC è quello di abbattere le differenze tra TAG e Reader. Inoltre i dispositivi operanti in modalità TAG, e che rispondono dunque alle interrogazioni della loro controparte, possono farlo sia in modalità operativa attiva, generando la potenza necessaria alla trasmissione, che in modalità passiva, cosi come fanno i TAG passivi. Il ruolo che svolge il dispositivo che inizia la comunicazione è denominato in particolare Initiator mentre quello giocato dal dispositivo interrogato è quello di Target. Quindi quando un Initiator intende iniziare una comunicazione col Target gli invia un messaggio, denominato nel gergo NFC message. Come conseguenza il Target, ricevuto il message, risponde tramite una opportuna risposta denominata invece reply.



Ruoli dei dispositivi NFC

In accordo al ruolo che ciascun dispositivo svolge nella comunicazione risulta ovvio che non tutte le modalità operative sono possibili come evidenziato nella seguente tabella

MODALITA OPERATIVA	RUOLO	
	Initiator	**Target**
Attiva	Possibile	Possibile
Passiva	Non Possibile	Possibile

A.2 Standard

Gli standard per i protocolli di comunicazione NFC presentano direttive analoghe in ISO, ECMA e parzialmente anche in ETSI. I principali sono

- **NFCIP-1** che descrive l'interfaccia radio, l'inizializzazione, l'anticollisione, un formato di trama e un protocollo di scambio dati a blocchi con gestione degli errori. Vengono definiti sia un modo di comunicazione Attivo che uno Passivo. Infine gli standard ETSI TS 102.190, ISO/IEC 18092 ed ECMA 340 definiscono, in maniera identica, l'NFCIP-1
- **NFCIP-2** il quale descrive un meccanismo di selezione tra differenti modi di comunicazione. Tramite suddetto meccanismo risulta possibile che gli stessi dispositivi possano comunicare sia secondo le modalità NFCIP-1 sia come Proximity Card ISO/IEC 14443. Viene quindi definita la modalità con cui un apparato NFCIP-2 compliant possa realizzare la comunicazione in una delle modalità descritte dai sopracitati standard. Gli standard ISO/IEC 21481 ed ECMA 352 descrivono in modo identico NFCIP-2. Si prevede inoltre la possibilità, in un prossimo futuro, di far comunicare i dispositivi NFC con altri dispositivi RFID operanti nella banda dei 13,56MHz come ad esempio i TAG passivi che vengono associati ai prodotti in vendita nei grandi magazzini
- **NFC–WI** che regola lo scambio dati, attraverso un interfaccia wired, tra dispositivo NFC e front-end dell'apparato che lo ospita che può essere un telefonino, un obliteratrice, un parchimetro e cosi via. Le specifiche includono i cavi di segnale, i segnali binari, i diagrammi di stato del protocollo e la codifica dei bit per tre diversi data rates. Gli standard ISO/IEC DIS 28361 ed ECMA 373 definiscono, in modo identico, l'NFC-WI

Esistono tuttavia parti dell'NFC non descritte dalle direttive ISO ed ECMA. Queste parti sono comunque definite da specifiche emanate dall'NFC Forum e riguardano prevalentemente i formati dei dati:

- **NFC Smart Poster Record Type Definition Technical Specification** nel quale è definito il modo di inserire dati ricevuti dall'esterno come URL, SMS, numeri di

telefono in un dispositivo NFC, nonché quello di trasferirli ad altro dispositivo. Definisce inoltre un formato di dati, lo Smart Poster Record Type Definition, utile in applicazioni in cui le informazioni sono associate agli oggetti

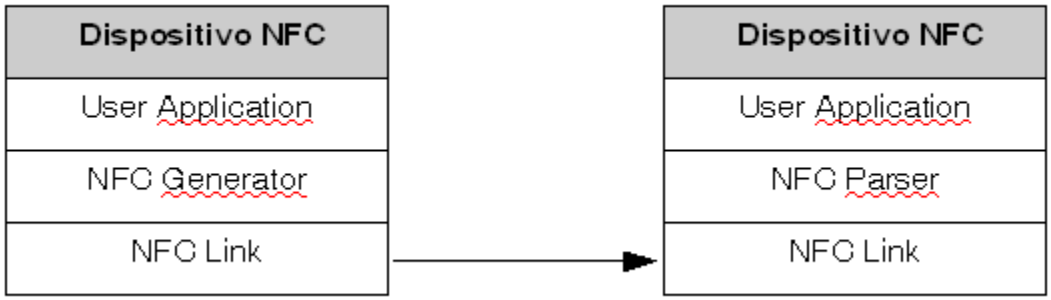
- **NFC Data Exchange Format Technical Specification** tramite cui viene descritto un formato, l'NDEF, per lo scambio dati tra dispositivi NFC. Contiene le regole per costruire un messaggio NDEF e definisce il meccanismo di incapsulamento dei dati all'interno dei messaggi
- **NFC Record Type Definition Technical Specification** che specifica i formati standard RTD per i messaggi NDEF, scambiati fra i dispositivi, al fine di assicurare l'interoperabilità della tecnologia
- **NFC Text RTD Technical Specification** che definisce la struttura delle informazioni per i dati dei testi, che possono essere usati per la descrizione di altri oggetti presenti all'interno del dispositivo. Il Text RTD può essere definito come un semplice campo di testo che aggiunge metadati a oggetti, come ad esempio gli URI
- **NFC URI RTD Technical Specification** nel quale è definita la struttura dei record atta a contenere gli URI. Questi descrivono risorse Internet che possono essere lette dagli NFC. Vengono inoltre definite le informazioni necessarie a trasportare una URI da un dispositivo NFC all'altro. L'URI RTD inoltre fornisce il modo per immagazzinare le URI dentro altri elementi NFC, come ad esempio gli Smart Posters

A.3 NDEF – NFC Data Exchange Format

Le specifiche NDEF descrivono un formato per l'incapsulamento dei dati all'interno di un singolo messaggio denominato appunto messaggio NDEF. Ciascun messaggio NDEF si compone di uno o più record NDEF, ciascuno contenente una certa quantità di dati, denominata in questo contesto PAYLOAD, che non può superare la massima dimensione di 2^{32}-1 byte. Quando il PAYLOAD ha una dimensione superiore a quella che un record può incorporare si ricorre al partizionamento del PAYLOAD. Le varie parti del PAYLOAD partizionato vengono dunque immagazzinate il altrettanti record NDEF che

in questo particolare contesto prendono il nome di *chunked record*.

La generazione dei messaggi NDEF avviene ad opera di un opportuno generatore NDEF il quale si occupa di inserire in uno o più record i dati sotto forma di PAYLOAD e successivamente di generare il messaggio integrandovi all'interno i record precedentemente costruiti. Fatto questo il messaggio è pronto per essere trasmesso verso un altro dispositivo NFC. Il messaggio, una volta giunto a destinazione, viene processato da un parser che si occupa di estrarre il PAYLOAD dai record all'interno del messaggio ed inviare i dati, eventualmente ricostruiti da un PAYLOAD partizionato, alla opportuna applicazione utente. La specifica applicazione a cui inviare i dati viene individuata tramite un opportuno campo ID presente nei record.



Trasmissione in un sistema NFC

A.3.1 Messaggi NDEF

Come già detto ciascun messaggio NDEF si compone di uno o più record NDEF il cui ruolo è quello di contenere il PAYLOAD. Come vedremo più avanti a ciascun record nel messaggio sono associati una serie di flag ed in particolare, per quel che interessa a noi ora, sono associati due particolari flag: MB (Message Begin) ed ME (Message End). Il primo record presente nel messaggio ha sempre settato il flag MB mentre l'ultimo ha settato il flag ME. Se un record ha entrambi i flag MB ed ME settati risulta evidentemente l'unico presente nel messaggio. Esiste inoltre la possibilità per un record NDEF di contenere un messaggio NDEF inglobandolo come PAYLOAD e realizzando cosi il cosiddetto nesting dei messaggi.

NDEF Message			
Record 1 MB=1, ME=0	Record 2 MB=0, ME=0	…..	Record n MB=0, ME=1

Rappresentazione di un messaggio NDEF

A.3.2 Record NDEF

Un record NDEF, come abbiamo già detto, rappresenta il contenitore del PAYLOAD all'interno del messaggio NDEF. Più in dettaglio si presenta costituito da un HEADER e dal PAYLOAD. L'HEADER è a sua volta strutturato in una serie di flag e campi indispensabili al parser per la gestione del PAYLOAD. Il PAYLOAD, come più volte detto, rappresenta invece il carico dati. La struttura di un record NDEF può essere rappresentata come di seguito.

NDEF Record								
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Bit**
MB	ME	CF	SR	IL	TNF			**Header**
TYPE_LENGTH								
PAYLOAD_LENGTH 3								
PAYLOAD_LENGTH 2								
PAYLOAD_LENGTH 1								
PAYLOAD_LENGTH 0								
ID_LENGTH								
TYPE								
ID								
PAYLOAD								**Payload**

I flag all'interno di un record NDEF sono i seguenti:

- MB, Message Begin, se settato indica che il record corrente rappresenta il primo record all'interno del messaggio NDEF
- ME, Message End, che se settato indica che il record corrente rappresenta l'ultimo record presente nel messaggio
- CF, Chunk Flag, se impostato ad uno indica che il record corrente è un *initial record chunk* oppure un *middle record chunk*
- SR ossia Short Record indica o meno se il record corrente ha il campo PAYLOAD_LENGTH costituito da un singolo byte. Questa caratteristica dei record NDEF si rivela molto utile nel caso di PAYLOAD di piccole dimensioni in cui non sono necessari quattro byte per codificare la lunghezza del PAYLOAD
- IL indica, se settato, la presenza del campo ID_LENGTH. Se impostato a zero allora nel record non sono presenti i campi ID_LENGTH e ID

I campi presenti all'interno di un record sono invece i seguenti

- TNF ovvero Type Name Format indica la struttura del valore impostato nel campo TYPE. Il campo è costituito da 3 bit e l'insieme dei possibili valori è indicato nella seguente tabella

Valore	TNF	Note
0x00	Vuoto	TYPE_LENGTH, ID_LENGTH, e PAYLOAD_LENGTH devono essere impostati a zero e TYPE, ID e PAYLOAD vengono omessi dal record
0x01	NFC Forum well-known type	
0x02	Media-Type definito nell'RFC 2046	
0x03	URI Assoluto definito nell'RFC 3986	
0x04	NFC Forum external type	
0x05	Sconosciuto	TYPE_LENGTH deve essere impostato a zero ed il campo TYPE non è presente

0x06	Unchanged ossia invariato rispetto al precedente *record chunk*	TYPE_LENGTH deve essere impostato a zero ed il campo TYPE non è presente
0x07	Riservato ad usi futuri	

- TYPE_LENGTH è un campo a 8 bit il cui scopo è di indicare la lunghezza in byte del campo TYPE
- ID_LENGTH con lunghezza di 1 byte, identifica la lunghezza in byte del campo ID ed è presente soltanto se il flag IL è settato. Se il suo valore è zero allora il campo ID non è presente
- PAYLOAD_LENGTH che indica la lunghezza in byte del PAYLOAD nel record. Se il flag SR è settato questo campo ha dimensione di un byte altrimenti di quattro byte. Inoltre se assume valore zero allora non è presente alcun PAYLOAD. Il valore massimo che può assumere è di 2^{32}-1 che corrisponde alla massima dimensione che può avere il PAYLOAD nel record
- TYPE con dimensione di 1 byte, serve ad indicare il tipo del PAYLOAD. La struttura e la codifica di questo campo sono in relazione con il valore impostato nel campo TNF
- ID formato da 8 bit, è un identificatore che si presenta nella forma di un riferimento URI, assoluto o relativo, come dalle specifiche RFC 3986. Il suo scopo è quello di fornire un identificativo del PAYLOAD presente nel record. Nel caso si presenti in forma di URI relativo la corretta gestione del riferimento è a carico dell'applicazione utente in quanto le specifiche NDEF non definiscono in alcun modo meccanismi per la definizione di URI base per gli URI relativi. Nel caso di PAYLOAD partizionati questo campo è presente soltanto nell'*initial record chunk*. C'è da dire infine che dell'unicità dell'ID si occupa il generatore del messaggio

A.3.3 Record Chunks

A volte può capitare che il PAYLOAD abbia dimensioni troppo grandi per essere contenuto in un singolo record. In questi casi si fa ricorso al partizionamento del PAYLOAD tramite il quale i dati da inserire in un messaggio vengono suddivisi in parti di

dimensioni tali da essere contenute in un record. Un record contenente una parte del PAYLOAD viene in questo caso denominato *record chunk*. Inoltre ciascun messaggio può contenere uno o più PAYLOAD partizionati. L'inserimento in un messaggio di un PAYLOAD partizionato viene realizzato

1. con l'inserimento della prima parte del PAYLOAD partizionato in un particolare *record chunk* denominato *initial record chunk*. L'*initial record chunk* è caratterizzato dal fatto che presenta il flag CF (Chunk Flag) settato. Inoltre i campi ID e TYPE vengono utilizzati per memorizzare l'identificatore ed il tipo dell'intero PAYLOAD partizionato mentre il campo PAYLOAD_LENGHT viene impiegato per identificare unicamente la dimensione dei dati contenuti nel record corrente e non dell'intero PAYLOAD
2. con l'inserimento delle parti successive del PAYLOAD partizionato, fatta eccezione per l'ultima, in altrettanti record denominati *middle record chunk* anch'essi con flag CF settati. I campi TYPE_LENGTH e IL vengono impostati a zero, il campo TNF (Type Name Format) con il valore 0x06 (Unchanged) ed infine il campo PAYLOAD_LENGHT con la dimensione dei dati nel *middle record chunk* corrente
3. inserendo infine l'ultima parte del PAYLOAD partizionato nel cosiddetto *terminating record chunk* avente flag CF impostato a zero. I campi TNF, TYPE_LENGHT, IL e PAYLOAD_LENGHT vengono impostati con gli stessi criteri usati nei *middle record chunk*

NDEF Message					
RECORD SINGOLO	**RECORD CHUNKS**				**RECORD SINGOLO**
Record 1 MB=1, ME=0, CF=0	Record 2 MB=0, ME=0, CF=1	Record 3 MB=0, ME=0, CF=1	…..	Record n-1 MB=0, ME=0, CF=0	Record n MB=0, ME=1, CF=0

Messaggio NDEF con payload partizionato

A.4 NFC Record Type Definition

Come abbiamo già avuto modo di vedere all'interno di ogni record NDEF è presente un particolare campo, il campo TYPE, che ha lo scopo di identificare il tipo, denominato *record type name*, del PAYLOAD. Esistono diversi formati con cui è possibile indicare il tipo del PAYLOAD, denominati *type name format*, ed il particolare formato utilizzato è codificato all'interno del campo TNF. I formati che possono essere indicati nel TNF sono NFC Forum well-known type, Media-type, URI assoluto, NFC Forum external type. Possono inoltre essere indicati altri valori quali il tipo sconosciuto, invariato (nel caso di *middle record chunk* e *terminating record chunk*) e vuoto.

A.4.1 NFC Forum well-known type

Rappresenta un formato breve per la descrizione del tipo del PAYLOAD. Viene usato quando non esistono URI o Media-type disponibili per indicare il tipo del PAYLOAD. Essendo inoltre un formato breve per la descrizione del tipo viene usato quando limitazioni di spazio impongono nomi di tipo molto brevi. L'uso del formato NFC Forum well-known type viene indicato impostando il campo TNF all'interno del record al valore 0x01. Inoltre il particolare tipo NFC Forum well-known type è rappresentato da un URN (Uniform Resource Name) come da specifiche RFC 2141. L'URN per l'NFC Forum well-known type presenta come Namespace Identifier “nfc” e il Namespace Specific String è sempre preceduto dal suffisso “wtk:”. All'interno del campo TYPE questo tipo viene però rappresentato sotto forma di URI relativo, come descritto nell'RFC 3986, da cui la definizione di formato breve. Quindi ad esempio l'URN “urn:nfc:wtk:miotipo” viene codificato all'interno del record NDEF come “miotipo”. Infine risulta importante distinguere tra NFC Forum Local Type, indicato sempre con l'iniziale del tipo minuscola, ed NFC Forum Global Type, con l'iniziale maiuscola. Lo scopo dell'NFC Forum Local Type è di identificare esclusivamente il tipo del PAYLOAD presente nel record corrente mentre quello dell'NFC Forum Global Type è di identificare il *record type name* di tutti quei record nei quali non è stato specificato il tipo.

A.4.2 NFC Forum external type

L'NFC Forum external type nasce dall'esigenza di organizzazioni esterne all'NFC Forum di avere uno spazio per i nomi di tipo specifico per le loro esigenze. L'utilizzo di questo formato per i *record type name* viene indicato settando il campo TNF al valore 0x04. Come nel caso degli NFC Forum well-known type anche qui il tipo è rappresentato da un URN dotato di Namespace Identifier pari a "nfc" ma con Namespace Specific String preceduto dal suffisso "ext". La parte restante del NSS è costituito dal nome di dominio dell'organizzazione che ha definito il il tipo più il nome del tipo vero e proprio. Tuttavia all'interno del record il nome del tipo viene indicato omettendo il NID ed il suffisso "ext" del NSS. Ad esempio il tipo "urn:nfc:ext:miaorganizzazione.com:miotipo" viene indicato nel campo TYPE come "miaorganizzazione.com:miotipo".

A.5 NFC Text Record Type Definition

I record di testo altro non sono che record in cui il PAYLOAD è rappresentato da un tipo NFC Forum well-known type. Il tipo del PAYLOAD per questi record è più precisamente indicato con "T" la cui codifica binaria NFC è 0x54. I record di testo sono stati concepiti con lo scopo di contenere al loro interno del testo semplice infatti la struttura del tipo è unicamente orientata a fornire informazioni sulla codifica del testo, UTF-8 o UTF-16, e sulla lingua in cui il testo è scritto. Il PAYLOAD in questi record si presenta strutturato come di seguito.

Offset in byte	Lunghezza in byte	Contenuto
0	1	Status byte
1	n	Codice lingua ISO/IANA
n+1	m	Testo codificato in UTF-8 oppure UTF-16

Denominati quindi

l la lunghezza complessiva del PAYLOAD in byte

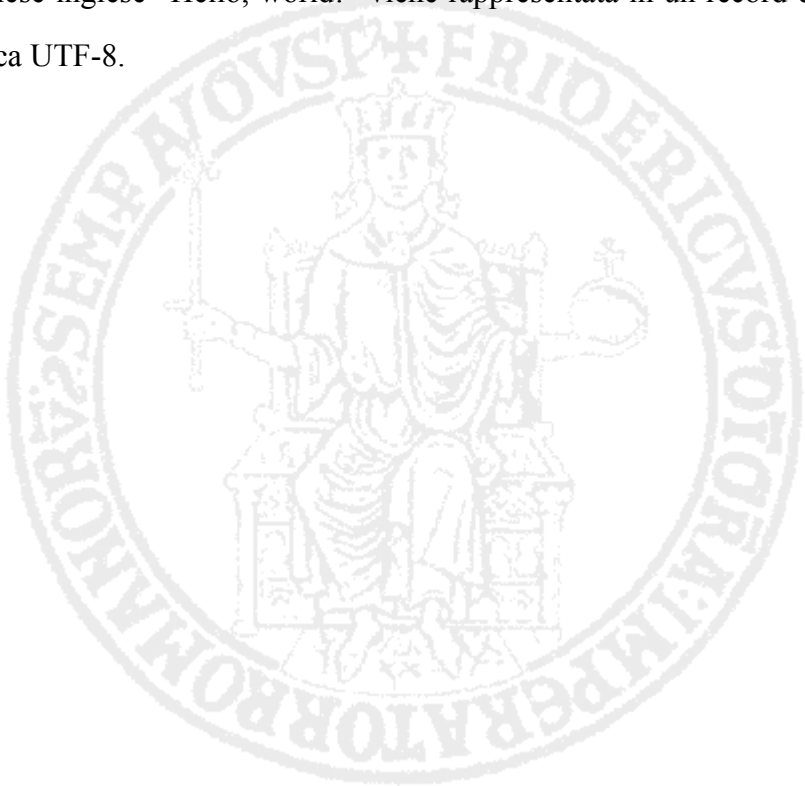
n la lunghezza del codice lingua in byte

la lunghezza del testo codificato *m* in byte è pari a $m = l - n - 1$

Lo status byte contiene informazioni ausiliarie necessarie all'interpretazione del testo codificato ed in particolare un bit necessario ad individuare la codifica del testo (UTF-8 oppure UTF-16) e 6 bit necessari ad indicare un codice identificativo della lingua utilizzata per il testo in accordo con l'RFC 3066. Il codice lingua si presenta composto da cinque o due caratteri a seconda che venga indicata o meno la variante della lingua. Se si vuole ad esempio specificare che il testo è in lingua inglese viene usata la sigla a due caratteri "en" composta da due caratteri mentre se si vuole indicare che il testo è in lingua inglese con variante statunitense si può far ricorso alla sigla "en-US" composta invece da cinque caratteri. Riportiamo di seguito un illustrazione della struttura dello status byte

Bit (0 è il bit meno significativo)	**Contenuto**
7	Si presenta pari a 0 se il testo è codificato in UTF-8, 1 se in UTF-16
6	Deve essere pari a 0
5...0	Lunghezza in byte del codice lingua

Per chiarezza nei riguardi di quanto detto fin ora mostriamo un esempio di come la frase in lingua inglese inglese "Hello, world!" viene rappresentata in un record di testo attraverso una codifica UTF-8.



NDEF Text Record			
Offset in byte	**Lunghezza in byte**	**Contenuto**	
0	1	SR=1, IL=0, TNF=0x01 (il valore del TNF indica che il tipo del PAYLOAD è un NFC Forum well-known type)	**Header**
1	1	TYPE_LENGTH=0x01 (infatti il nome del tipo, “T”, ha lunghezza pari ad un byte)	
2	1	PAYLOAD_LENGTH=0x10	
3	1	TYPE="T"	
4	1	STATUS_BYTE=0x02 (infatti la codifica è in UTF-8 e il codice lingua è di due caratteri)	**Payload**
5	2	CODICE_LINGUA="en"	
7	13	Codifica in UTF-8 di "Hello, world!"	

A.6 NFC URI Record Type Definition

I record URI sono particolari tipi di record NDEF in cui il tipo del PAYLOAD è un NFC Forum well-known type. Più in particolare il tipo del PAYLOAD viene indicato con “U” avente codifica binaria NFC pari a 0x55. Scopo di questi tipi di record è la rappresentazione degli URI (Uniform Resource Identifier). La struttura del PAYLOAD di un record URI è illustrata di seguito.

Offset in byte	Lunghezza in byte	Contenuto
0	1	Codice identificativo URI
1	n	Campo URI

Detta quindi l la lunghezza complessiva del PAYLOAD in byte è possibile calcolare la lunghezza del Campo URI n in byte come $n = l - 1$

Il codice identificativo URI ha lo scopo di codificare il campo protocollo presente all'interno di un URI cosi da ridurre il più possibile la dimensione complessiva dell'URI da rappresentare. Il campo URI invece serve a contenere la parte di un URI che esclude il campo protocollo. Volendo quindi includere un URL del tipo "http://www.ingegneria-informatica.unina.it" in un record URI basta indicare nel Codice identificativo URI il codice corrispondente a "http://www." e nel Campo URI la codifica in UTF-8 della parte rimanente dell'URL ovvero "ingegneria-informatica.unina.it". L'insieme dei valori che può assumere il Codice Identificativo URI in relazione all'insieme dei possibili protocolli è riportato nella tabella di seguito.

Valore decimale	Valore esadecimale	Protocollo
0	0x00	Nessuno
1	0x01	http://www.
2	0x02	https://www.
3	0x03	http://
4	0x04	https://www.
5	0x05	tel:
6	0x06	mailto:
7	0x07	ftp://anonymous:anonymous@
8	0x08	ftp://ftp.
9	0x09	ftps://
10	0x0A	sftp://
11	0x0B	smb://
12	0x0C	nfs://
13	0x0D	ftp://
14	0x0E	dav://
15	0x0F	news:
16	0x10	telnet://
17	0x11	imap:
18	0x12	rtsp://
19	0x13	urn:
20	0x14	pop:

21	0x15	sip:
22	0x16	sips:
23	0x17	tftp:
24	0x18	btspp://
25	0x19	btl2cap://
26	0x1A	btgoep://
27	0x1B	tcpobex://
28	0x1C	irdaobex://
29	0x1D	file://
30	0x1E	urn:epc:id:
31	0x1F	urn:epc:tag:
32	0x20	urn:epc:pat:
33	0x21	urn:epc:raw:
34	0x22	urn:epc:
35	0x23	urn:nfc:
36…255	0x24..0xFF	RFU ossia trattati come per 0x00

Da notare che ogniqualvolta che nel campo Codice Identificativo URI compaiono i valori esadecimali 0x00 e uno di quelli compresi nell'intervallo 0x24 e 0xFF il Codice Identificativo URI viene trascurato. Infatti in questo caso si suppone che l'URI da rappresentare sia completamente specificato nel Campo URI.

Facciamo a questo punto un esempio per riassumere quanto detto fin ora supponendo di voler definire un record URI contenente l'URL “http://www.nfc.com”. Il record in questione si presenta come di seguito.

NDEF URI Record			
Offset in byte	**Lunghezza in byte**	**Contenuto**	
0	1	SR=1, IL=0, TNF=0x01 (il valore del TNF indica che il tipo del PAYLOAD è un NFC Forum well-known type)	**Header**
1	1	TYPE_LENGTH=0x01 (infatti “U” ha lunghezza pari ad un byte)	
2	1	PAYLOAD_LENGTH=0x08	
3	1	TYPE="U"	
4	1	URI_Identifier_Code=0x01	**Payload**
5	7	Codifica in UTF-8 di "nfc.com"	

A.7 NFC Smart Poster Record Type Definition

L'idea alla base degli Smart Poster Record è quella che un dispositivo NFC può memorizzare informazioni su se stesso esattamente come fanno i TAG. In questo modo quando un dispositivo NFC tocca il dispositivo TAG in questione risulta in grado di leggere suddette informazioni. Gli Smart Poster Record sono anche in grado di memorizzare informazioni circa le azioni che un dispositivo NFC che legge il dispositivo TAG deve intraprendere una volta effettuata la lettura.

Gli Smart Poster Record comunque non sono altro che dei particolari tipi di record in cui il PAYLOAD è di un tipo NFC Forum well-known type identificato dalla sigla "Sp" la cui codifica binaria NFC è 0x53, 0x70. Più in dettaglio il PAYLOAD si presenta, in questi record, come un messaggio NDEF costituito da uno o più record NDEF. I tipi di record NDEF all'interno del PAYLOAD degli Smart Poster Record sono tutti degli NFC Forum well-known type ed in particolare Title Record, URI Record, Action Record, Icon Record

e Type Record.

A.7.1 URI Record

Rappresentano la base degli Smart Poster Record infatti tutti gli altri eventuali record presenti nel PAYLOAD fungono semplicemente da metadati per questo record. Risulta importante capire che ci può essere un solo record URI all'interno del PAYLOAD.

A.7.2 Title Record

Si tratta di un istanza del tipo NFC Text Record e la sua presenza nel PAYLOAD è opzionale. Inoltre possono essere presenti più di uno di questi record ognuno dei quali contiene testo in una lingua differente. Risulta comunque consentita la presenza di un solo Title Record per ogni lingua.

A.7.3 Raccomanded Action Record

Questo particolare record descrive in che modo dovrebbe essere trattato il servizio che il TAG NFC mette a disposizione. Ad esempio potrebbe contenere una direttiva avente lo scopo di indicare al dispositivo NFC lettore di aprire in un browser l'URI indicato nel record URI. La sua presenza resta comunque opzionale infatti se non presente il dispositivo NFC lettore può autonomamente decidere quale azione intraprendere mentre la sua presenza rappresenta null'altro che un suggerimento per il dispositivo NFC lettore. Il campo d'azione di questo tipo di record è di tipo locale infatti il tipo del suo PAYLOAD è rappresentato da un NFC Forum Local Type identificato dalla sigla "act" e con rappresentazione binaria NFC 0x61, 0x63, 0x74. Infine notiamo che il PAYLOAD in questi record è rappresentato da un solo byte che può assumere uno dei valori elencati nella seguente tabella.

Valore	Azione Corrispondente
0x00	Intraprendi l'azione
0x01	Salva
0x02	Apri per la modifica
0x03...0xFF	RFU

Il valore 0x00 indica di intraprendere un azione che a seconda dei casi può essere l'invio di un messaggio SMS, l'avvio di un browser, l'avvio di un telefonata. Il valore 0x01 suggerisce a seconda dei casi di effettuare il salvataggio di un SMS, di inserire un numero telefonico in un elenco contatti, ecc. Il valore 0x02 suggerisce invece di intraprendere un azione corrispondente alla modifica della risorsa indicata nel record URI quindi ad esempio la modifica di un numero di telefono o quella di un SMS. Infine l'insieme dei valori che vanno da 0x03 a 0xFF non indicano nessuna azione specifica e vengono pertanto ignorati.

A.7.4 Icon Record

Si tratta di un record opzionale il cui scopo è quello di contenere immagini corrispondenti a diversi tipi MIME. In uno Smart Poster Record possono essere contenuti più di un Icon Record ciascuno dei quali ingloba un immagine in un formato diverso. Tramite questo record viene data la possibilità al dispositivo lettore di selezionare e dunque visualizzare una sola di queste immagini in base alle proprie possibilità di visualizzazione.

A.7.5 Size Record

Si tratta anche in questo caso di un record opzionale. Il suo scopo è quello di identificare la dimensione dell'oggetto referenziato dal record URI e quindi di informare preventivamente il lettore NFC su cosa aspettarsi all'apertura dell'URI. Il PAYLOAD in questi record è di un tipo NFC Forum Local Type identificato con "s" ed è composto da quattro byte.

Byte 0	Byte 1	Byte 2	Byte 3

Struttura del PAYLOAD di un Size Record

A.7.6 Type Record

Record opzionale il cui PAYLOAD è costituito da una stringa codificata in UTF-8. Il suo scopo è quello di descrivere il tipo dell'oggetto referenziato in un record URI. Infine il PAYLOAD è di un tipo NFC Forum Local Type ed in particolare quello identificato dalla stringa "t".

A.7.7 Esempi di Smart Poster Record

Chiariamo quanto detto fin ora con degli esempi di Smart Poster Record. Il primo riguarda la rappresentazione di uno Smart Poster Record contenente un semplice record URI in cui è contenuto l'indirizzo web "http://www.nfc-forum.org"

NDEF Smart Poster Type Record				
Offset in byte	**Lunghezza in byte**	**Contenuto**		
0	1	MB=1, ME=1, SR=1, IL=0, TNF=0x01 (il valore del TNF indica che il tipo del PAYLOAD è un NFC Forum well-known type)	**Header**	
1	1	TYPE_LENGTH=0x02 (infatti "Sp" ha lunghezza pari a due byte)		
2	1	PAYLOAD_LENGTH=0x12 (la lunghezza dei dati dello Smart Poster Record è pari a 18 byte)		
3	2	TYPE="Sp"		
5	1	MB=1, ME=1, SR=1, TNF=0x01	**URI Record**	**Payload**
6	1	TYPE_LENGTH=0x01		
7	1	PAYLOAD_LENGTH=0xE (0xE equivale a 14 byte)		
8	1	TYPE="U"		
9	1	URI_Identifier_Code=0x01 (0x01 corrisponde al protocollo "http://www.")		
10	13	Codifica in UTF-8 di "nfc-forum.org"		

Di seguito un altro esempio di record Smart Poster Type in cui questa volta sono presenti un record URI, contenente l'indirizzo web "http://www.nfc-forum.org", ed uno di tipo Action nel quale è suggerito di aprire l'URI con un browser.

NDEF Smart Poster Type Record				
Offset in byte	**Lunghezza in byte**	**Contenuto**		
0	1	MB=1, ME=1, SR=1, IL=0, TNF=0x01 (il valore del TNF indica che il tipo del PAYLOAD è un NFC Forum well-known type)	**Header**	
1	1	TYPE_LENGTH=0x02 (infatti "Sp" ha lunghezza pari a due byte)		
2	1	PAYLOAD_LENGTH=0x12 (la lunghezza dei dati dello Smart Poster Record è pari a 18 byte)		
3	2	TYPE=0x53 (0x53 è la codifica binaria NFC di "Sp")		
5	1	MB=1, ME=0, SR=1, TNF=0x01	**URI Rec**	**Payload**
6	1	TYPE_LENGTH=0x01		
7	1	PAYLOAD_LENGTH=0xE (0xE equivale a 14 byte)		
8	1	TYPE="U"		
9	1	URI_Identifier_Code=0x01 (0x01 corrisponde al protocollo "http://www.")		
10	13	Codifica in UTF-8 di "nfc-forum.org"		
11	1	MB=0, ME=1, SR=1, TNF=0x01	**Action Rec**	
12	1	TYPE_LENGTH=0x03		
13	1	PAYLOAD_LENGTH=0x01		
14	3	TYPE="act"		
17	1	0x00		

Appendice B

La Crittografia

La crittografia rappresenta l'arte di progettare algoritmi, denominati in questo caso cifrari o in inglese cypher, tramite la quale risulta possibile trasformare il testo di un messaggio, denominato testo in chiaro, rendendolo incomprensibile a tutti tranne al suo destinatario. Oltre che ad opera dell'algoritmo di cifratura, il cifrario, la cifratura avviene per mezzo di un ulteriore parametro: la chiave di cifratura. La decifratura di un messaggio cifrato da parte del destinatario viene invece realizzata ad opera di un algoritmo simile a quello utilizzato dal mittente per cifrare il messaggio e di un parametro denominato chiave di decifratura che a seconda dell'algoritmo utilizzato per cifrare il messaggio può coincidere o meno con la chiave di cifratura. La sicurezza di un sistema di crittografia risiede esclusivamente nella segretezza della chiave di decifratura e non dell'algoritmo che è opportuno far conoscere alla pubblica analisi, in modo che se ne possano scoprire eventuali punti deboli in tempo.

La crittografia ha origini molto antiche: gli ebrei utilizzavano un cifrario denominato atbash, gli spartani avevano un loro particolare sistema di comunicazione dei messaggi segreti denominato scitala e viene attribuita a Gaio Giulio Cesare l'invenzione del cosiddetto Cifrario di Cesare con il quale ogni lettera del messaggio veniva sostituita con un altra, più avanti di un certo numero di posizioni. Inoltre nel 1466 Leon Battista Alberti pubblicò un suo libro in cui descriveva i principali metodi di cifratura conosciuti all'epoca e introduceva una nuova tecnica inventata personalmente che consisteva in una sostituzione simile a quella di Cesare con sostituzione periodica della chiave.

E' possibile considerare un cifrario come una funzione che si serve di due parametri per generare il testo cifrato: il testo in chiaro e la chiave di cifratura. In maniera assolutamente simile è possibile vedere un algoritmo per la decifratura ossia come una funzione che opera sulla base del testo cifrato e della chiave di decifratura al fine di restituire il testo in

chiaro. Le operazioni di cifratura e decifratura possono dunque essere riassunte come segue:

```
Mc = C( M, EncKey )
M = D( Mc, DecKey )
```

dove `C` rappresenta l'algoritmo di cifratura

`D` rappresenta l'algoritmo di decifratura

`M` è il messaggio da cifrare

`Mc` è il messaggio cifrato

`EncKey` è la chiave di cifratura

`DecKey` è la chiave per la cifratura

La lunghezza della chiave utilizzata è uno dei fattori più importanti per la segretezza del testo evitando infatti che possa essere decifrato per tentativi. Infatti provare tutte le possibili combinazioni di caratteri che potrebbero formare una chiave è un problema che gli analisti definiscono a complessità computazionale esponenziale dato che l'aggiunta anche di un solo carattere alla chiave aumenta in modo vertiginoso il numero di possibili combinazioni che possono essere ottenute.

Gli scopi con cui viene utilizzata la crittografia sono essenzialmente di

1. garantire la **riservatezza** del messaggio in modo che soltanto il destinatario sia in grado di estrarre dal testo cifrato il testo in chiaro
2. garantire l'**integrità** del messaggio consentendo quindi al destinatario di verificare che il messaggio pervenutogli non sia stato alterato
3. consentire l'**autentificazione** del mittente ossia consentire al destinatario di identificare il mittente e verificare che il presunto mittente abbia realmente inviato il messaggio
4. garantire la **non ripudiazione** di un messaggio da parte del mittente in modo da impedire al mittente di negare di avere spedito il messaggio

B.1 La crittografia simmetrica

Le tecniche di crittografia di tipo simmetrico si basano su algoritmi che operano la cifratura e la decifratura servendosi di un unica chiave. La riservatezza delle informazioni contenute in un messaggio è quindi legata alla riservatezza di suddetta chiave di cui devono essere a conoscenza unicamente il mittente ed il destinatario del messaggio. La tutela della riservatezza della chiave rappresenta quindi il motivo per cui nella crittografia simmetrica la chiave, condivisa tra mittente e destinatario, viene denominata chiave privata. Riassumendo, la cifratura e la decifratura mediante algoritmi a crittografia simmetrica possono essere illustrate come di seguito.

```
Mc = C( M, PrivKey )
M = D( Mc, PrivKey )
```

dove `C` rappresenta l'algoritmo di cifratura

`D` rappresenta l'algoritmo di decifratura

`M` è il messaggio da cifrare

`Mc` è il messaggio cifrato

`PrivKey` è la chiave privata per la cifratura e la decifratura

B.1.1 Il Cifrato di Cesare

Tramite questa tecnica ogni lettera nel messaggio viene sostituita con un altra, più avanti nell'ordine alfabetico di un certo numero di posizioni. In questo caso il numero di posizioni successive alla lettera da sostituire rappresenta la chiave di cifratura. Supponendo quindi di voler effettuare la cifratura del seguente messaggio, composto con le lettere dell'alfabeto anglosassone

```
MIO MESSAGGIO
```

diventa, scegliendo una chiave di cifratura pari a 3

```
PNR PHVVDJJLT
```

Ovviamente affinché il messaggio possa essere successivamente letto è necessario che il

ricevente abbia a disposizione la chiave per la decifratura che risulta essere la stessa utilizzata per cifrare il messaggio. Questa volta però anziché sostituire ogni lettera con quella più avanti di un numero di posizioni pari alla chiave si deve sostituire con la lettera più indietro del numero di posizioni pari alla chiave.

Una variante su suddetto cifrario è quella di avere ogni lettera del testo in chiaro sostituita con qualche altra lettera in modo autonomo e senza una legge fissa. Un sistema di questo tipo è detto a sostituzione monoalfabetica e la chiave è rappresentata dalla stringa composta dal numero di lettere di cui si compone l'alfabeto con cui è composto il messaggio. Tramite questo metodo ed adoperando la seguente chiave

```
QAZWSXEDCRFVTGBYHNUJMIKOLP
```

ogni lettera di cui si compone un messaggio viene sostituita adoperando la corrispondenza evidenziata dalla seguente tabella

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Q	A	Z	W	S	X	E	D	C	R	F	V	T	G	B	Y	H	N	U	J	M	I	K	O	L	P

In questo modo scegliendo la precedente chiave è possibile cifrare il seguente messaggio

```
MIO MESSAGGIO
```

ottenendo

```
TCB TSUUQEECB
```

Anche in questo caso, come nel Cifrato di Cesare, il destinatario del messaggio deve essere in possesso della stessa chiave utilizzata per la cifratura per riuscire a decifrare il testo ricevuto.

B.1.2 Il Cifrario di Leon Battista Alberti

Si tratta di una tecnica di cifratura molto simile a quella adottata da Cesare ma in questo caso vengono adottate chiavi differenti per cifrare le differenti parole che compongono il

messaggio. Ad esempio il messaggio

```
MIO MESSAGGIO
```

viene cifrato, adottando la chiave costituita dalla coppia (3, 4), nel seguente modo

```
PNR QIWWEKKMS
```

In questo modo il messaggio può essere decifrato sostituendo ogni lettera della prima parola con la lettera che la precede di 3 posizioni nell'ordine alfabetico e ad ogni lettera della seconda con la lettera che la precede di 4 posizioni.

B.1.3 Il codice di Vigènere

Il codice di Vigenère si basa un'operazione che viene chiamata sostituzione polialfabetica, molto più sicura di una semplice sostituzione monoalfabetica. Il Vigenère propose l'uso della tavola quadrata, composta da alfabeti ordinati spostati di una lettera. La tavola è composta dalla lista decifrante scritta orizzontalmente in testa mentre le liste cifranti sono solo le 25 sottostanti ciascuna individuata dalla loro prima lettera che fa parte della chiave per cifrare e decifrare. La tavola si presenta come di seguito:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M

O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

La cifratura di un certo messaggio viene praticata scegliendo innanzitutto la chiave di cifratura composta da una sequenza di lettere ad esempio GERRY. Fatto questo è necessario estrarre dalla precedente tabella oltre alla prima riga quelle righe le cui iniziali corrispondono alle lettere di cui si compone la chiave quindi nel nostro caso

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X

Giunti a questo punto andremo a sovrapporre il messaggio da cifrare con una ripetizione della chiave come mostrato nel seguente esempio

```
MIO MESSAGGIO
GER RYGERRYGE
```

e successivamente rimpiazzare ogni lettera del messaggio con la corrispondente cifrata nell'incrocio tra la colonna individuata dalla lettera del messaggio (quelle della prima riga) e la linea individuata dalla lettera chiave. Il messaggio oggetto d'esempio verrà quindi

cifrato come segue

```
MIO MESSAGGIO
SMF DCYWRXEOS
```

La forza di questo cifrato sta nel fatto che il numero di chiavi è enorme quindi gli attacchi praticati ad esempio provando tutte le possibili combinazioni non sono praticabili. Per quasi trecento anni rimase infatti inviolato ma poi cadde di fronte all'analisi del colonnello prussiano Friedrich Kasiski, che nel 1863 pubblicò un libro che conteneva un metodo di decrittazione della tavola di Vigenère e del suo cifrario polialfabetico con chiave ripetuta.

B.1.4 Il cifrario a trasposizione per colonne

I cifrari visti fino a questo momento conservano l'ordine dei simboli in chiaro e si limitano a trasformarli in modo più o meno complesso. Il cifrario a trasposizione per colonne si basa invece su una tecnica chiamata trasposizione, in cui le lettere dell'alfabeto non vengono trasformate ma semplicemente ordinate secondo determinati criteri. In questo cifrario lo scopo della chiave è numerare le colonne di una tabella che contiene il testo in chiaro scritto orizzontalmente per righe della lunghezza della chiave. Il testo cifrato deve invece essere letto per colonne, seguendo l'ordine alfabetico proposto dalla chiave. Illustriamo questa tecnica supponendo di voler cifrare il messaggio

```
MIO MESSAGGIO
```

sfruttando la chiave

```
KEY
```

Procederemo quindi alla cifratura del nostro messaggio utilizzando una tabella in cui la prima riga è composta dalle lettere della chiave e le successive dalle lettere componenti il messaggio da cifrare. Nel nostro caso suddetta tabella si presenta come segue

K	E	Y
M	I	O
M	E	S
S	A	G
G	I	O

Il risultato della cifratura sarà dato dalla lettura delle colonne della precedente tabella, seguendo l'ordine alfabetico delle lettere che compongono la chiave che nel nostro caso è EKY. Il risultato della cifratura sarà quindi

```
IEAIMMSGOSGO
```

La decifratura del messaggio cifrato può essere invece realizzata creando una nuova tabella il cui, come nel caso precedente, la prima riga è composta dalle lettere di cui è formata la chiave ed il numero di colonne è determinato dalla lunghezza del messaggio cifrato. Si procede quindi a riempiendo la tabella per colonne con il messaggio cifrato e leggendo infine le righe successive alla prima.

B.1.5 La macchina Enigma

L'ultimo passo prima della così detta crittografia moderna è costituito dalla costruzione dalla macchina elettromeccanica tedesca ENIGMA usata nella seconda guerra mondiale. Essa era composta da ruote con i caratteri incisi sul bordo, e con contatti elettrici in corrispondenza delle lettere in entrambi i lati. Il testo in chiaro, digitato su una tastiera, veniva riprodotto utilizzando i caratteri della prima ruota, la quale a sua volta costruiva un nuovo alfabeto utilizzando i caratteri della seconda, e poi della terza, e cosi' via. Tutte le ruote, e potevano essere parecchie, venivano "scalate", in modo che la sostituzione delle lettere fosse ogni volta diversa. La chiave consisteva nel settaggio iniziale delle ruote, che potevano essere posizionate in una quantità di posizioni diverse tanto alta quante più erano le ruote utilizzate. Questo meccanismo è facile da costruire via software e abbastanza sicuro, può tuttavia essere infranto. Fu brillantemente attaccato dal matematico polacco Marin Rejewsky che con il suo lavoro permise di decifrare numerosi messaggi militari

tedeschi, un fattore che probabilmente contribuì alla vittoria finale degli alleati.

B.1.6 Il DES

Il DES (Data Encryption Standard) rappresenta un cifrario inizialmente sviluppato dall'IBM e successivamente modificato dalla National Security Agency (NSA). Fu adottato dal governo statunitense nel 1977 ufficialmente per la protezione di dati riservati ma non classificati come segreti militari o di stato e che tuttora è usato da tutte le agenzie federali, fatta eccezione per quegli atti che richiedevano un livello più alto di sicurezza.

Il DES è un codice cifrato a blocchi. La chiave usata per cifrare è un blocco di 64 bit suddivisa in 8 sottoblocchi di 8 bit ciascuno. L'ultimo bit di ogni sottoblocco è di controllo quindi i bit che costituiscono effettivamente la chiave sono 56. Il testo da cifrare viene suddiviso in blocchi di 64 bit ciascuno e vengono cifrati uno dopo l'altro in successione con uguale procedimento. Se un blocco non raggiunge la lunghezza desiderata di 64 bit si utilizza un procedimento detto *pad*, che può essere implementato in diversi modi: un metodo aggiunge zeri fino alla lunghezza stabilita mentre un altro, se i dati sono binari, integra il blocco con bit che sono l'opposto degli ultimi bit del messaggio. Nel caso di dati ASCII si usano invece byte generati in modo casuale specificando nell'ultimo byte il carattere ASCII corrispondente al numero di byte aggiunti. Infine un'ultima tecnica, in parte equivalente alla precedente, usa sempre bit casuali ma fornisce, negli ultimi tre bit, il numero di byte originali, cioè quelli che costituiscono il messaggio senza riempimento. Giunto a questo punto l'algoritmo si articola come di seguito

1. viene innanzitutto cambiata la posizione di ciascun blocco di testo di 64 bit. Viene dunque alterato l'ordine originale dei blocchi di 64 bit che compongono il testo in chiaro
2. ciascun blocco di 64 bit viene a sua volta suddiviso in due blocchi da 32 bit ciascuno
3. per ciascuno dei blocchi di 32 bit vengono effettuati 16 passaggi i quali si articolano in una serie di sostituzioni e trasposizioni in base a delle sottochiavi ricavate a partire dalla chiave originale

4. terminati i 16 passaggi di cui sopra i due blocchi di 32 bit, il cui contenuto è stato alterato, vengono riuniti

Più in dettaglio ciascuno dei 16 passaggi, tramite i quali vengono alterati i blocchi di 64 bit di cui si compone il messaggio, può essere a sua volta illustrato come segue. Indicati con

`T(i)`	il risultato dell'i-esimo passaggio
`S(i)`	il blocco di 32 bit sinistro all'i-esimo passaggio
`D(i)`	il blocco di 32 bit destro durante l'i-esimo passaggio
`KEY(i)`	la sottochiave utilizzata nell'i-esimo passaggio
`AND`	l'operazione di *and* logico bit a bit
`XOR`	l'operazione di *or* esclusivo bit a bit

avremo

1. `T(i) = S(i) AND D(i)`
2. `S(i) = D(i-1)`
3. `D(i) = S(i-1) XOR f( D(i-1), KEY(i) )`

Occupiamoci adesso di studiare in che modo opera la funzione *f()* che compare nell'ultimo dei precedenti tre passaggi. La funzione opera tramite i seguenti passaggi

1. il blocco `D(i-1)` viene espanso da 32 bit a 48 ossia `E[ D(i-1) ]`
2. viene calcolato `E[ D(i-1) ] XOR K(i)`
3. quanto ottenuto al precedente passaggio ossia un blocco di 48 bit viene suddiviso in 8 blocchi di 6 bit ciascuno che indicheremo con `D(1), D(2),…, D(8)`
4. ciascun blocco `B(i)` viene posto in ingresso ad una funzione che indicheremo con *Z* la quale restituisce stringhe di 4 bit ossia `Z[ B(i) ]`. La funzione *Z* opera nel seguente modo: preleva da ogni matrice fissata S-Box (Substitution Box) i 4 bit del nuovo blocchetto `S(i) = Z[ B(i) ]` posizionati in base alle righe e colonne specificate dai 6 bit del corrispondente `B(i)`
5. gli 8 blocchetti `S(1), S(2), …, S(8)` vengono concatenati e poi scambiati di posto ottenendo alla fine `P[ S(1),S(2), …, S(8)] = f[ D(i-1), KEY(i) ]`

Infine non ci resta altro che esaminare in che modo vengono calcolate le sottochiavi `K(i)`

utilizzate dalla funzione *f* per il calcolo di `f[ D(i-1), KEY(i) ]`. Come detto in precedenza la chiave per la cifratura e la decifratura è una stringa di 64 bit con 8 bit di controllo che vengono ignorati durante le operazioni di cifratura e decifratura. La chiave si presenta quindi effettivamente costituita da 56 bit che vengono suddivisi in due sottoblocchi di 28 bit, che denotiamo con `S(0)` e `D(0)`. Dopodiché per 16 volte i sottoblocchi vengono spostati a sinistra ottenendo `S(1)`, `D(1)`, `S(2)`, `D(2)`, ..., `S(16)`, `D(16)`. Quindi al primo passo l'algoritmo utilizzerà la sottochiave `KEY(1) = P[ S(1) AND D(1) ]` dove abbiamo indicato con *P* una trasposizione (lo scambio di posto), al secondo `KEY(2) = P[ S(2) AND D(2)]` ed infine al sedicesimo `KEY(16) = P[ S(16) AND D(16) ]`. In questo modo tutte le operazioni effettuate producono sottochiavi `K(i)` di 48 bit.

La decifratura avviene in maniera pressoché identica all'operazione di cifratura con la differenza differenza che le sottochiavi utilizzate in ogni passo sono diverse. In particolare al primo passo viene utilizzata `K(16) = P[ L(16) AND L(16) ]`, al secondo `K(15) = P[ L(15) AND L(15) ]` e cosi via.

Il DES fu considerato un algoritmo di cifratura sicuro fino al 17 Luglio 1998 quando la Electronic Frontier Foundation diffuse un comunicato stampa nel il quale annunciava la possibilità di cracking dell'algoritmo DES.

B.1.7 L'algoritmo IDEA

IDEA (International Data Encryption Algorithm) nasce nel 1991 in Svizzera sotto il nome di IPES (Improved Proposed Encryption Standard) ad opera di famosi ricercatori: Xuejja Lai e James L. Massey. Esattamente come il DES è un codice cifrato a blocchi di 64 bit da cui differisce per il fatto che la chiave è di 128 bit, caratteristica che dovrebbe eliminare qualsiasi possibilità di riuscita di ricerca della chiave procedendo per tentativi, poiché l'insieme delle possibili chiavi è di 2^128. La cifratura con IDEA comporta una divisione del blocco di 64 bit del testo in chiaro in 4 sottoblocchi di 16 bit. Ogni sottoblocco subisce 8 passi in cui sono coinvolte 52 sottochiavi diverse a 16 bit ottenute dalla chiave a 128 bit. Più in dettaglio viene adottato il seguente procedimento al fine di generare le sottochiavi:

1. la chiave a 128 bit viene suddivisa in 8 blocchi di 16 bit che costituiscono le prime 8 sottochiavi
2. le cifre della chiave a 128 sono spostate di 25 bit a sinistra in modo da generare una nuova combinazione, il cui raggruppamento ad 8 bit fornisce le prossime 8 sottochiavi
3. il passaggio precedente viene ripetuto fin quando non sono state generate tutte le 52 sottochiavi

Ogni passo comporta calcoli abbastanza semplici come XOR, addizione e moltiplicazioni in modulo 16 (significa che i risultati non possono superare i 16 bit quindi quelli eccedenti vengono scartati). Durante gli 8 passi il secondo e il terzo blocco si scambiano di posto mentre all'ultimo passo i 4 sottoblocchi vengono concatenati per produrre un blocco di testo cifrato a 64 bit.

La decodifica è identica eccetto il fatto che le sottochiavi sono ottenute in maniera diversa dalla chiave principale a 128.

IDEA è al momento il cifrario a chiave segreta più utilizzato quanto riguarda i software commerciali di crittografia vista la sua velocità di codifica e decodifica e la sua elevata sicurezza.

B.2 La crittografia asimmetrica

Le tecniche di cifratura esaminate fino ad ora utilizzano un unica chiave sia per l'operazione di cifratura che di decifratura. Un simile sistema comporta problemi legati alla riservatezza della chiave utilizzata la cui conoscenza deve essere dominio esclusivo del mittente e del ricevente del messaggio. Ad esempio un primo problema che si presenta è rappresentato dallo scambio della chiave tra mittente e ricevente che deve avvenire tramite un canale di comunicazione sicuro. Risulta ostico trovare una soluzione ad un simile problema nel momento in cui si vuole effettuare uno scambio di messaggi su lunghe distanze. Ecco quindi che nel 1975 Whitfield Diffie e Martin Hellman idearono un sistema per la cifratura e decifratura dei messaggi denominato asimmetrico. Diffie ed Hellman pensarono ad un sistema basato sull'uso di due chiavi generate in modo che sia impossibile

ricavarne una dall'altra. Le due chiavi vengono chiamate pubblica e privata: la prima serve per cifrare e la seconda per decifrare. Una persona che deve comunicare con un'altra persona non deve far altro che cifrare il messaggio con la chiave pubblica del destinatario il quale una volta ricevuto il messaggio non deve fare altro che decifrarlo con la chiave segreta personale. Utilizzando un simile sistema ogni persona possiede una coppia di chiavi: quella pubblica può essere tranquillamente distribuita e resa di pubblico dominio perché consente solo di cifrare il messaggio mentre quella privata deve essere conosciuta solo da una persona.

B.2.1 L'RSA

Una prima implementazione delle tecniche di crittografia asimmetrica fu sviluppata nel 1978 da Ronald Rivest, Adi Shamir e Leonard Adleman che realizzarono una procedura di calcolo che prese il nome di algoritmo RSA.

L'algoritmo RSA si basa su un procedimento che, sfruttando i numeri primi, risulta estremamente improbabile attaccare. Dati due numeri primi è molto facile stabilire il loro prodotto mentre risulta molto più difficile determinare, a partire da un determinato numero, quali numeri primi hanno prodotto quel risultato dopo essere stati moltiplicati tra loro. La complessità delle operazioni necessarie a determinare la chiave privata partendo dalla chiave pubblica risulta pertanto troppo elevata per essere messa effettivamente in pratica. La procedura che illustra l'algoritmo si articola nei seguenti passi:

1. partendo da due numeri primi, `p` e `q`, viene calcolato il loro prodotto `n = p*q`
2. calcolo del valore `z = (p-1)*(q-1)`
3. scelta di un intero `d` tale che `d` che sia primo rispetto a `z`, il che significa che i due numeri non devono avere fattori primi in comune
4. ricerca di un numero `e` tale che `e*d mod z = 1`, cioè che il resto della divisione tra `e*d` e `z` sia 1
5. suddivisione del messaggio in chiaro in blocchi costituiti da `k` bit, dove `k` è il più grande intero che soddisfa la disequazione `2^k < n`
6. per ciascun blocco di `k` bit che chiameremo `m` si procede con la cifratura calcolando

`Mc = m^e mod n` mentre la decifratura avviene calcolando `Mc^d mod n`

Dal procedimento esposto di sopra si evince che per cifrare un messaggio in chiaro è necessario conoscere `e` ed `n` che sono quindi parte costituente della chiave pubblica mentre per decifrare il testo cifrato è necessario conoscere `d` ed `n` che sono invece parte della chiave privata.

B.3 Sistemi crittografici ibridi

Nonostante i sistemi crittografici asimmetrici riescano a garantire la riservatezza nello scambio dei messaggi risultano estremamente lenti, a differenza dei sistemi simmetrici, sia nella fase di cifratura dei messaggi che nella decifratura. Risulta pertanto oneroso riuscire a mettere in piedi una comunicazione cifrata basata su di un sistema a cifratura asimmetrico. Sono stati quindi ideati sistemi per lo scambio sicuro di messaggi che sfruttano in maniera combinata le tecniche di cifratura simmetrica ed asimmetrica. In un simile sistema un algoritmo asimmetrico, quale l'RSA, viene utilizzato unicamente per lo scambio della chiave privata tra le due controparti tra le quali deve avvenire la comunicazione. Successivamente la cifratura dei messaggi avviene invece sfruttando un algoritmo di cifratura simmetrico quale ad esempio il DES oppure l'IDEA. Possiamo riassumere, denominate A e B le entità tra le quali deve avvenire lo scambio di messaggi, come segue

1. A utilizza la chiave pubblica di B per cifrare la chiave di sessione ed invia il risultato dell'operazione a B
2. B utilizza la sua chiave privata per decifrare quanto ricevuto da A ed ottenendo la chiave di sessione
3. da questo momento in poi le controparti coinvolte possono scambiarsi messaggi, cifrati con un algoritmo simmetrico, in maniera sicura

B.4 Funzioni di hash

Una funzione di hash è una funzione non invertibile il cui scopo è quello di convertire un testo di lunghezza arbitraria, ad esempio il testo di un messaggio, in una stringa di

lunghezza fissa e relativamente limitata. La stringa ottenuta può essere considerata come una sorta di impronta digitale del testo in chiaro e viene detta *valore di hash*, *checksum crittografico* o *message digest*. Non essendoci una corrispondenza biunivoca tra il valore di hash e il testo in base al quale il valore di hash è stato calcolato e dato che l'insieme dei testi possibili ha cardinalità maggiore di quella dell'insieme degli hash possibili, per il principio dei cassetti ad almeno un hash corrisponderanno più testi possibili. Quando, a partire da due testi differenti, viene prodotto lo stesso hash, si parla di *collisione* e la qualità di una funzione di hash è misurata direttamente in base alla difficoltà nell'individuare due testi che generino una collisione. Le funzioni di hash possono essere utilizzate con diversi scopi quali ad esempio

1. la verifica dell'integrità di un messaggio trasmesso
2. generazione di una firma digitale

La verifica dell'integrità di un messaggio viene realizzata dal lato trasmettitore inviando assieme al messaggio il *checksum crittografico*, calcolato in base al messaggio da inviare, e dal lato ricevitore calcolando il *checksum crittografico* del messaggio ricevuto e confrontandolo successivamente con quello spedito dal trasmettitore assieme al messaggio.

Importantissimo, come vedremo in seguito, risulta il contributo che gli algoritmi di hash forniscono del processo di generazione della firma digitale. Le funzioni di hash, infatti, permettono la rapida creazione della firma anche per file di grosse dimensioni. Risulterebbe infatti computazionalmente sconveniente generare una firma, tramite l'esecuzione quindi di algoritmi di cifratura asimmetrici, sulla base di una grande mole di dati mentre molto più conveniente è invece generare una firma sulla base del *checksum crittografico*, calcolato sulla base del messaggio che si intende cifrare.

Tra gli algoritmi di hash ricordiamo l'MD2, l'MD4 e l'MD5, considerati oramai insicuri essendo stati trovati metodi per generare collisioni in tempi relativamente brevi, ed i moderni SHA-1 e SHA-2.

B.5 La firma digitale

Fino ad ora, tramite la descrizione di vari algoritmi di cifratura, abbiamo centrato l'attenzione sulla ricerca di metodi che consentissero all'utente di preservare la riservatezza del messaggio. A volte però non sussiste la necessità di cifrare un intero messaggio ma si desidera semplicemente avere un modo per riuscire a stabilire l'identità dell'autore del messaggio stesso. A questo proposito ci viene in contro la firma digitale tramite la quale siamo tranquillamente in grado di risolvere i problemi relativi

1. all'**integrità** del messaggio
2. all'**autentificazione** del mittente del messaggio
3. all'impossibilità del mittente di un messaggio di **ripudiare** quanto da lui firmato

La firma digitale si basa sull'utilizzo combinato di algoritmi di cifratura asimmetrici e delle cosiddette funzioni di hash. Come visto nel paragrafo dedicato alla crittografia asimmetrica, la chiave privata di un certo utente è la sola in grado di poter decifrare correttamente i dati cifrati con la chiave pubblica di quell'utente. In suddetto paragrafo non abbiamo però detto che un testo cifrato con una certa chiave privata può essere decifrato con la chiave pubblica associata. Proprio quest'ultimo rappresenta il principio su cui si basa la firma digitale. In maniera più rigorosa il processo di creazione di una firma digitale si articola nei seguenti passi:

1. si ricava, tramite l'ausilio di un algoritmo di hash (come l'MD5 oppure l'SHA-1) che deve essere reso noto al destinatario del documento, il *message digest* relativo al documento da firmare
2. si procede col cifrare, tramite un algoritmo crittografico asimmetrico (ad esempio l'RSA o il DSA) da rendere noto anch'esso al destinatario, con la chiave privata del mittente il *message digest* ottenuto al passo precedente ottenendo quindi la firma digitale relativa al documento da firmare.

Il documento unitamente alla firma digitale, calcolata come spiegato di sopra, rappresenta un documento firmato digitalmente che può essere letto da chiunque in quanto composto da testo in chiaro ma non modificato poiché la sua alterazione verrebbe evidenziata dalla firma digitale ad esso allegata. In particolare il destinatario del documento può controllare

che il documento ricevuto non sia stato alterato

1. decifrando la firma allegata al documento con la chiave pubblica del mittente ottenendo quindi un *message digest* corrispondente presumibilmente al documento ricevuto
2. calcolando localmente il *message digest* relativo al documento e confrontandolo poi con quello ricavato al passo precedente. Se i due coincidono è possibile concludere che il documento non è stato, con una probabilità molto alta, alterato in quanto l'algoritmo di hash è fatto in modo da minimizzare la probabilità che da testi diversi si giunga allo stesso *message digest*.

Appendice C

Java Me, JSR 257 e Boucy Castle Cryptography API

Più che un linguaggio di programmazione Java rappresenta una vera e propria piattaforma per lo sviluppo di software. Ideata da James Goslin ed altri dipendenti della Sun Microsystems questa piattaforma ha come obbiettivo la possibilità di creare applicazioni in grado di essere eseguite su dispositivi dalle caratteristiche hardware anche moto differenti tra loro. Java si presenta pertanto composta da un linguaggio di programmazione, una *Java Virtual Machine* ed un insieme di API e tool per lo sviluppo delle applicazioni. Una particolare rilevanza assume la *Java Virtual Machine*, in breve JVM, lo strumento tramite il quale viene resa possibile non solo la semplice portabilità del codice ma la portabilità di intere applicazioni. Infatti dal codice sorgente Java, tramite un processo di compilazione, viene generato il cosiddetto *bytecode* il quale potrà successivamente essere eseguito da una qualsiasi implementazione della JVM, secondo le specifiche rilasciate in un documento denominato *Java Virtual Machine* Specification, la quale si occuperà di tradurre le istruzioni specificate nel *bytecode* in chiamate al sistema operativo sottostante.

Nell'ambito del nostro studio risulta importante affrontare un discorso relativo ai dispositivi elettronici con risorse hardware limitate come cellulari e PDA. Infatti le limitazioni in termini di risorse di questi dispositivi impongono la definizione di ambienti di sviluppo ed esecuzione delle applicazioni differenti da quelli che si potrebbero utilizzare invece per sistemi dotati di elevate capacità di elaborazione. A tal scopo è stata dunque concepita la piattaforma *Java 2 Micro Edition*, in breve J2ME.

C.1 La piattaforma Java 2 Micro Edition

La piattaforma J2ME si presenta come un ambiente per lo sviluppo, il deploy e l'esecuzione di applicazioni Java in dispositivi dotati di risorse limitate. La definizione di un simile ambiente significa definire

- una JVM in grado di eseguire *bytecode* sul dispositivo nella quale è installata
- un insieme di API che permettano di sfruttare le caratteristiche del dispositivo
- un insieme di tool per lo sviluppo e il deploy delle applicazioni

La definizione di un suddetto ambiente comporta però problemi legati alle diversità e limitazioni dei dispositivi in commercio. Nasce pertanto l'esigenza di definire un modo per classificare i dispositivi non in base al loro utilizzo ma in base alle loro caratteristiche. Sono stati definiti a questo scopo i concetti di *Configuration* e *Profile*.

C.1.1 Configuration

Una *Configuration* è una descrizione completa di un *Java Runtime* per una categoria di dispositivi dotati di caratteristiche hardware simili. Essa si può considerare costituita da

- una JVM per l'esecuzione del *bytecode*
- un insieme di classi standard
- un insieme di classi per l'interfacciamento col sistema operativo al fine di sfruttare le caratteristiche peculiari del dispositivo

Mediante la definizione di una *Configuration* è possibile quindi specificare quali sono le operazioni minime che una JVM può eseguire e quali sono le API, presentate sotto forma di classi, disponibili per lo sviluppo delle applicazioni. Attualmente per la piattaforma J2ME sono definite due C*onfiguration*: il *Connected Device Configuration* (CDC) e il *Connected Limited Device Configuration* (CLDC). Le specifiche relative al CDC e al CLDC sono definite in documenti denominati *Java Specification Request*, in breve JSR. In particolare le specifiche del CDC, giunte oramai alla versione 1.0, sono definite nel JSR 36 mentre quelle inerenti il CLDC, giunte alla versione 1.1, sono definite nel JSR 139.

In particolare le specifiche della CDC riguardano dispositivi con caratteristiche hardware quali

- CPU a 32 bit
- Almeno 512K di memoria ROM
- Almeno 256K di memoria RAM
- Alimentazione teoricamente illimitata

- Connettività continua verso un qualche tipo di rete
- Capacità di gestione di un implementazione completa della JVM come dalle specifiche del *Java Virtual Machine Specification 2nd Edition*
- Interfaccia grafica di vario genere se non addirittura assente

Le specifiche della CLDC riguardano invece dispositivi dotati di

- CPU a 16 o 32 bit
- Alimentazione a batterie
- Almeno 192K di memoria complessiva
- Connettività limitata
- Interfaccia grafica di vario tipo se non addirittura assente

A questa categoria di dispositivi appartengono dispositivi come PDA e telefoni cellulari.

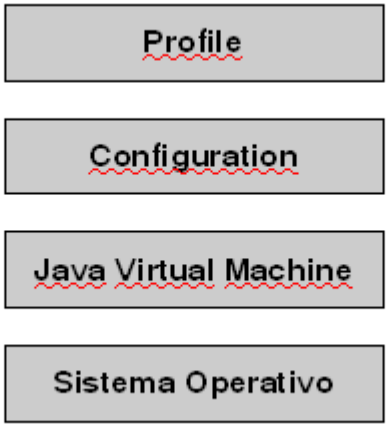
C.1.2 Profile

Mentre una *Configuration* consente di classificare un dispositivo in base alle caratteristiche hardware un *Profile* consente una classificazione dei dispositivi in base alle funzionalità che questi offrono. Un *Profile* si presenta quindi costituito da un insieme di API, che si poggiano su di una determinata *Configuration*, in grado di sfruttare le particolari caratteristiche del dispositivo. I *Profile* attualmente disponibili per la CDC sono

- il *Foundation Profile*
- il *Personal Basis Profile*
- il *Personal Profile*

mentre attualmente l'unico *Profile* disponibile per la CLDC è il *Mobile Information Device Profile*, in breve MIDP. Di seguito ci occuperemo del MIDP le cui specifiche, giunte alla versione 2.1, sono definite nel JSR 118.

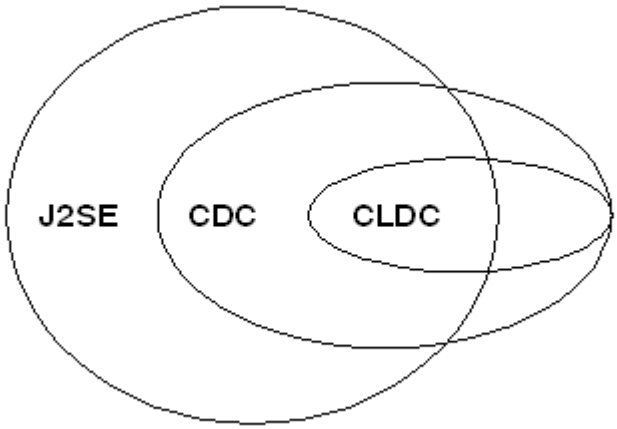
In base a quanto detto finora, saremo in grado di fornire una rappresentazione dell'architettura della J2ME come nella figura di seguito.



Architettura della piattaforma Java 2 Micro Edition

C.1.3 Il CLDC

Abbiamo visto in precedenza quali sono i requisiti hardware minimi che i dispositivi che implementano le CLDC devono avere. Si tratta di dispositivi elettronici di largo consumo come PDA e telefoni cellulari dotati di capacità di elaborazione esigua. Pertanto il *Java Runtime* disponibile per questo genere di dispositivi non può essere paragonato a quello messo a disposizione invece dai dispositivi che implementano le specifiche della CDC. Ad esempio CDC prevede che un dispositivo che la implementa abbia la capacità di gestire un implementazione completa della JVM. Al contrario la CLDC prevede la definizione di una versione semplificata della JVM, denominata invece KVM. Un discorso analogo può essere fatto per quanto riguarda le API messe a disposizione dal CLDC le quali sono costituite da un sottoinsieme di quelle disponibili invece per il CDC. Una *Configuration* quale il CLDC si può dunque considerare costruita a partire dal CDC tramite l'eliminazione di tutte quelle caratteristiche non necessarie oppure la cui implementazione richiederebbe la presenza di risorse che non sono invece presenti. Un discorso analogo può essere fatto riguardo la *Configuration* CDC, ricavata invece dalla piattaforma *Java 2 Standard Edition*. La relazione tra J2SE, CDC e CLDC è messa in evidenza dalla seguente figura in cui viene mostrato come il CDC sia definita come un sottoinsieme della J2SE con l'aggiunta di alcune estensioni. La CLDC può essere vista invece come un semplice sottoinsieme della CDC.



Relazione tra J2SE, CDC e CLDC

Per quanto riguarda le API messe a disposizione da questa *Configuration* possiamo distinguerle in due gruppi: classi ereditate dalla J2SE e classi specifiche della CLDC. Le classi ereditate dalla J2SE sono raccolte nei *package*

- `java.lang`
- `java.lang.ref`
- `java.util`
- `java.io`

mentre le classi esplicitamente create per la CLDC sono raggruppate nel *package*

- `java.microedition.io`

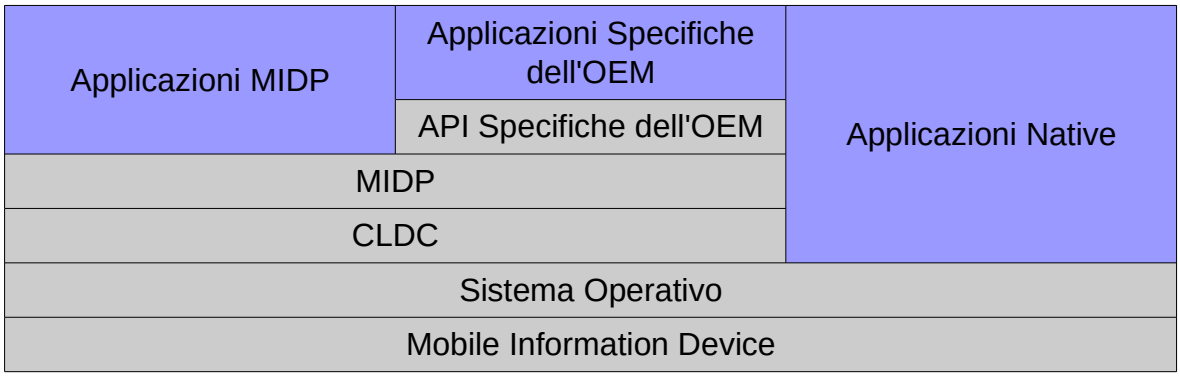
C.1.4 Il MIDP

Il MIDP, nell'implementazione descritta nel JSR 118, descrive un architettura ed un insieme di API per lo sviluppo di applicazioni in un ambiente per dispositivi definiti MID, *Mobile Information Device*. Le specifiche del MIDP riguardano principalmente aspetti legati a

- *Delivery* delle applicazioni e tecnologie per il *billing* delle stesse
- Gestione del ciclo di vita delle applicazioni
- Modello di firma delle applicazioni e definizione dei domini di sicurezza
- *MIDlet Push Registration* per la realizzazione di servizi di *Push*
- Gestione del N*etworking*

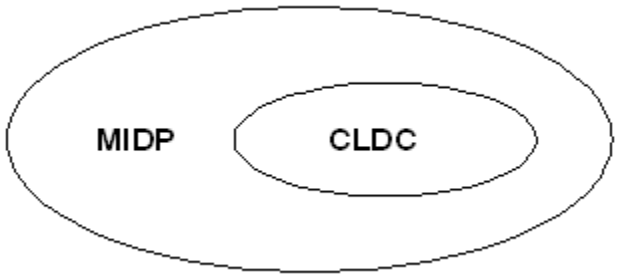
- Gestione della persistenza
- Gestione di contenuti multimediali
- Supporto alla realizzazione di interfacce grafiche

L'architettura di un sistema che soddisfa le specifiche dell MIDP è schematizzata di seguito



Architettura di un sistema con MIDP

Il MIDP descrive un insieme di API per lo sviluppo di applicazioni che prendono il nome di *Midlet*. L'insieme delle API che il MIDP descrive rappresenta un estensione di quello definito dal CLDC sul quale si poggia. Al programmatore di applicazioni basate sul MIDP vengono quindi rese disponibili sia le API descritte dal CLDC che quelle esplicitamente definite dal MIDP.



Relazione tra API del Profile MIDPe quelle

Le API del MIDP si presentano inoltre costituite da un insieme di classi raggruppate nei *package*

- `java.lang`
- `java.lang.ref`

- `java.util`
- `java.io`
- `javax.microedition.io`
- `javax.microedition.midlet`
- `javax.microedition.lcdui`
- `javax.microedition.game`
- `javax.microedition.media`
- `javax.microedition.media.control`
- `javax.microedition.rms`
- `javax.microedition.pki`

C.2 Le Midlet

Come già detto in precedenza un applicazione basata sul Profile MIDP è denominata *Midlet*. Una o più *Midlet*, all'interno di un archivio sotto forma di file *.jar*, compongono quella che viene denominata *Midlet Suite*. A supporto della *Midlet Suite* viene in genere fornito un file denominato *Java Application Descriptor*, in breve JAD, il cui scopo è quello di fornire informazioni relative alla *Midlet Suite*. All'interno del JAD, più in particolare, sono presenti tutta una serie di informazioni che possono essere classificate come

- *Required*
- *Optional*
- *Custom*

Riportiamo di seguito in tabella quelle che sono le informazioni di tipo *Required*.

Informazioni di tipo Required	Descrizione
MIDlet-Name	Il nome che identifica la Midlet Suite
MIDlet-Version	Versione della Midlet Suite
MIDlet-Vendor	Identificatore del rivenditore della Midlet Suite
MIDlet-Jar-Size	Indica la dimensione in byte del file .jar

MIDlet-Jar-URL	Rappresenta l'URL che identifica il .jar
MicroEdition-Configuration	Identifiica quale dev'essere la Configuration supportata dal dispositivo per poter eseguire le applicazioni nella Midlet Suite
MicroEdition-Profile	Profile che il dispositivo deve supportare per eseguire le applicazioni nella Midlet Suite

Riportiamo anche l'elenco delle possibili informazioni di tipo *Optional* che possono essere inserite nel JAD.

Informazioni di tipo Optional	**Descrizione**
MIDlet-Description	Rappresenta una descrizione testuale delle funzionalità dell'applicazione
MIDlet-Icon	È il percorso di un file immagine che rappresenta la Midlet Suite relativamente al file .jar che la contiene
MIDlet-Info-URL	URL identificativo di informazioni aggiuntive relative alla Midlet Suite
MIDlet-Data-Size	Permette di specificare la quantità minima di byte che la Midlet Suite richiede per il RMS il cui valore di default è zero
MIDlet-Install-Notify	Rappresenta un URL a cui il dispositivo può notificare l'avvenuta installazione di una Midlet Suite
MIDlet-Delete-Confirm	Rappresenta un URL a cui il dispositivo può notificare l'avvenuta accettazione di un operazione di cancellazione di una Midlet Suite
MIDlet-Delete-Notify	Rappresenta un URL a cui il dispositivo può notificare l'avvenuta cancellazione di una Midlet Suite

Per motivi di spazio non ci occuperemo delle informazioni di tipo *Custom* ma passeremo direttamente all'esame della struttura di un applicazione *Midlet*.

C.2.1 Struttura di una MIDlet

Una applicazione *Midlet* non rappresenta altro che un istanza di un estensione della classe astratta *MIDlet*, definita nel *package javax.microedition.midlet*. La classe *MIDlet* prevede infatti la presenza di tre metodi astratti e con visibilità *protected* che una classe derivata deve implementare:

- `protected void startApp()`
- `protected void pauseApp()`
- `protected void destroyApp(boolean unconditional) throws MIDletStateChangeException`

Tutti e tre i metodi sopra citati sono coinvolti, come vedremo in seguito, nel ciclo di vita di una *Midlet*. Inoltre anche il costruttore si presenta con visibilità *protected*. Infine è da notare che una classe che estende la classe *MIDlet* deve implementare al più il solo costruttore di default in quanto rappresenta il primo contatto con l'AMS.

C.2.2 Ciclo di vita di una MIDlet

Un dispositivo che soddisfa le specifiche del MIDP deve necessariamente implementare un applicazione che viene identificata in maniera indifferente con i termini di *Application Management Software* (AMS) oppure *Java Application Manager*. Compito fondamentale dell'AMS è quello di gestire il ciclo di vita di un applicazione *Midlet*. Una *Midlet* infatti può trovarsi in diversi stati in relazione alla sua esecuzione che sono

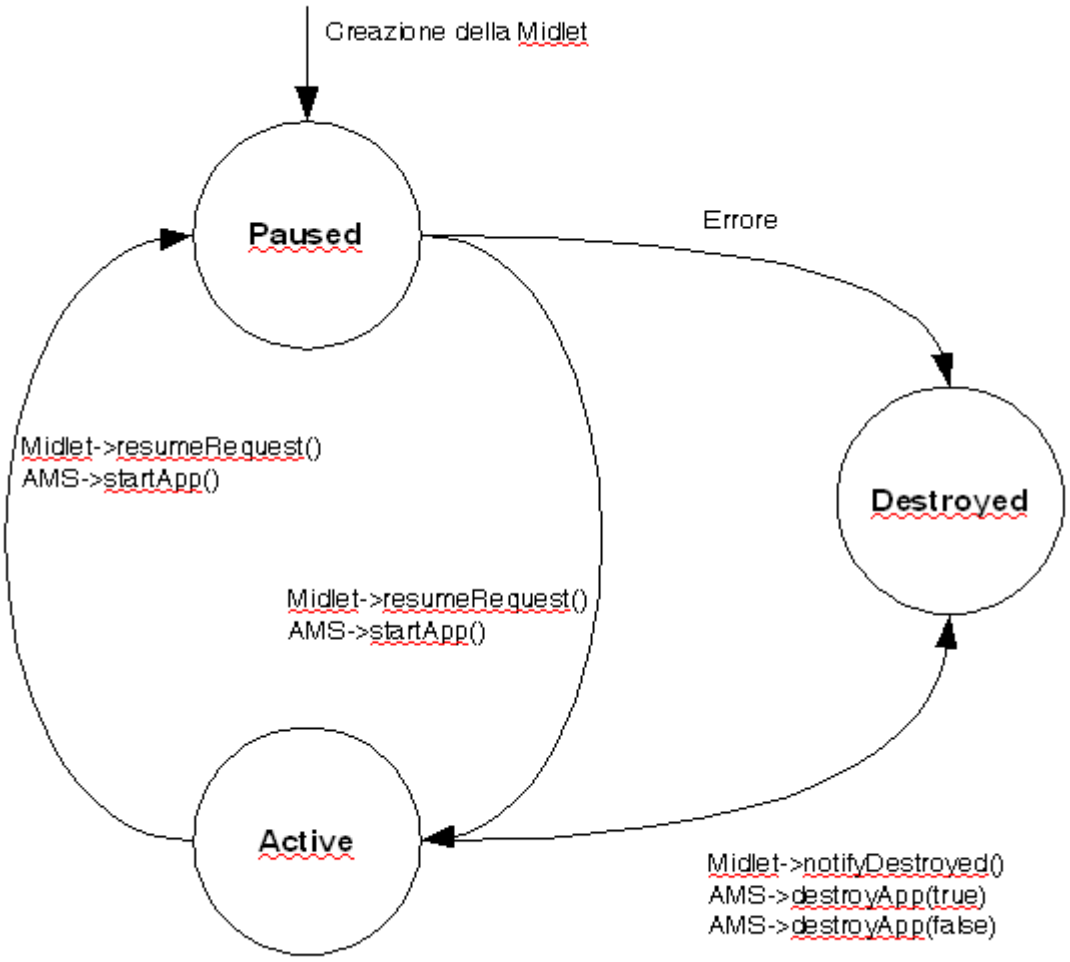
- *Active* nel quale una *Midlet* è in esecuzione nell'ambiente di esecuzione Java
- *Paused* nel quale la *Midlet* è in uno stato di sospensione ed ha liberato risorse eventualmente acquisite in precedenza
- *Destroyed* quando la *Midlet* ha terminato la sua esecuzione ed ha liberato tutte le risorse

Una *Midlet* si trova nello stato di *Active* quando l'AMS ne ha autorizzato l'esecuzione creando un istanza della classe che la descrive ed invocando successivamente il metodo *startApp()* il quale contiene tutte le istruzioni necessarie alla *Midlet* per acquisire le risorse necessarie alla sua esecuzione.

Quando una *Midlet* si trova nello stato di *Paused* significa che la sua esecuzione è stata interrotta per una precisa volontà dell'utente oppure a causa di una decisione dell'AMS. Questo è lo stato in cui si trova l'applicazione immediatamente dopo la creazione di un istanza della classe che la descrive. Nel caso in cui non venga sollevata nessuna eccezione durante l'esecuzione del codice nel costruttore di default, l'AMS, con l'invocazione del metodo *startApp()*, porta l'applicazione nello stato di *Active*. Esistono tuttavia altri motivi per cui una *Midlet* può trovarsi in stato di *Paused* ad esempio per iniziativa dell'AMS oppure dell'utente che utilizza l'applicazione. Nel primo caso ad esempio ciò può avvenire a seguito dell'arrivo di una telefonata. L'AMS sospenderà quindi l'esecuzione per permettere all'utente di rispondere. Al termine della telefonata sarà lo stesso AMS a provvedere al ripristino dell'applicazione precedentemente messa in pausa. Il meccanismo in base al quale viene notificato alla *Midlet* di entrare in stato di *Paused* si basa sulla presenza del metodo *pauseApp()* che contiene tutte le istruzioni necessarie affinché la *Midlet* liberi le risorse impegnate nel corso della propria esecuzione. Quindi quando l'AMS decide di mettere in pausa una *Midlet* lo fa invocando su di essa il metodo *pauseApp()*. È tuttavia possibile che la richiesta di ingresso nello stato di *Paused* avvenga ad opera della *Midlet* stessa. Questo avviene mediante l'invocazione, all'interno della *Midlet*, del metodo statico *notifyPaused()*. Risulta importante sottolineare che l'invocazione di questo metodo per notificare all'AMS che la *Midlet* è in stato di *Paused* non implica una successiva chiamata da parte dell'AMS del metodo *pauseApp()*. Sarà quindi compito della *Midlet* liberare tutte le risorse impegnate priva di invocare il metodo *notifyPaused()*.

Infine avremo che una *Midlet* si trova nello stato di *Destroyed* quando ha rilasciato tutte le risorse ed ha terminato la sua esecuzione. Come nel caso del passaggio allo stato di *Paused*, ci sono due possibili motivi per cui una *Midlet* può entrare nello stato di *Destroyed*: per una decisione dell'AMS oppure per volontà della stessa *Midlet*. L'AMS richiede alla *Midlet* il passaggio allo stato di *Destroyed* invocando sull'applicazione stessa il metodo *destroyApp(boolean unconditional)*. Quando il valore del parametro booleano è *true* ed il metodo viene eseguito senza sollevare eccezioni l'applicazione viene terminata

regolarmente. Quando invece il valore del parametro è *false* allora la *Midlet* può esprimere o meno un consenso alla sua terminazione tramite il sollevamento di un eccezione del tipo *MIDletStateChangeExcception* definita nel *package javax.microedition.midlet*. Infatti se il parametro booleano assume valore *false* ed il metodo *destroyApp(boolean unconditional)* solleva un eccezione di tipo *MIDletStateChangeExcception* l'AMS può eventualmente decidere di rimandare la terminazione dell'applicazione. Infine, come già accennato, la richiesta di passaggio allo stato *Destroyed* può anche arrivare dalla *Midlet*. Questo avviene mediante invocazione, all'interno del corpo del programma, del metodo *notifyDestroyed()*. In quest'ultimo caso sarà responsabilità dell'applicazione deallocare le risorse impegnate in precedenza.



Transizioni di stato in una Midlet

C.2.3 Esempio di MIDlet

Vediamo ora un esempio di quanto detto fin ora. Di seguito possiamo vedere il codice dell'applicazione che abbiamo chiamato *HelloWorldMIDlet*.

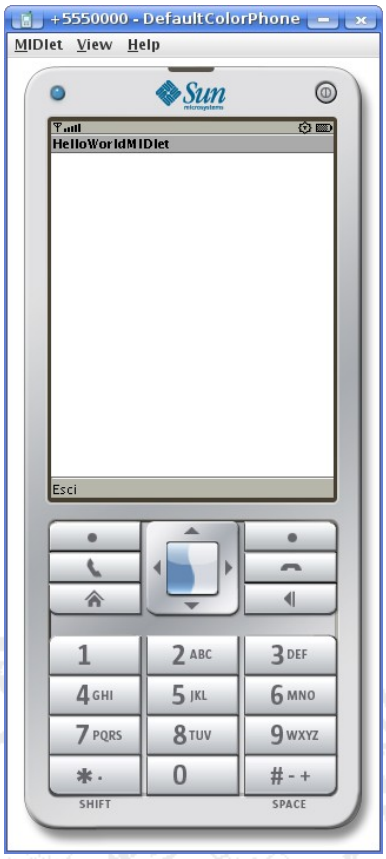
```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;


public class HelloWorldMIDlet extends MIDlet implements
        CommandListener {

      public HelloWorldMIDlet() {}

      protected void destroyApp( boolean arg0 ) throws
             MIDletStateChangeException {}

      protected void pauseApp() {}

      protected void startApp() throws MIDletStateChangeException {

            Form displayable = new Form( MIDLET_TITLE );
            Command exitCommand = new Command( EXIT_LABEL,
                                               Command.EXIT, 1 );
            displayable.addCommand( exitCommand );
            displayable.setCommandListener( this );
            Display.getDisplay( this ).setCurrent( displayable );

      }

      public void commandAction( Command arg0, Displayable arg1 ) {

            if ( EXIT_LABEL.equals( arg0.getLabel() ) ) {
                  try { destroyApp( true ); }
                  catch ( MIDletStateChangeException e ) {
                        e.printStackTrace();
                  }
                  notifyDestroyed();
            }

      }

      private final String MIDLET_TITLE = "HelloWorldMIDlet";
      private final String EXIT_LABEL = "Esci";

}
```

Possiamo invece ora vedere il contenuto del *Java Application Descriptor*

```
MIDlet-1: HelloWorldMIDlet,,HelloWorldMIDlet
MIDlet-Jar-Size: 1424
MIDlet-Jar-URL: HelloWorldMIDlet.jar
```

```
MIDlet-Name: HelloWorldMIDlet MIDlet Suite
MIDlet-Vendor: Gerardo Saracino
MIDlet-Version: 1.0.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Infine ecco come si presenta la nostra *Midlet* in esecuzione all'interno di un emulatore



Midlet in esecuzione all'interno di un emulatore

C.3 Le Contactless Communication API

Le API definite nel JSR 257 (Java Specificaiton Request 257), denominate Contactless Communication API, costituiscono un insieme di API per l'accesso ai TAG RFID e i codici a barre. Oltre alle classi e alle interfacce per l'accesso ai generici TAG RFID, vengono inoltre definite apposite strutture per interagire con i TAG contenenti dati formattati secondo le specifiche dell'NFC Forum, con le smart card compatibili con lo

standard ISO 14443 e con i codici a barre. Le strutture sopra citate sono strutturate in cinque package:

- `javax.microedition.contactless` nel quale vengono definite una serie di classi ed interfacce comuni a tutti i tipi di Target supportati dalle API
- `javax.microedition.contactless.ndef` per lo scambio dati con i TAG con dati formattati secondo le specifiche NFC Forum
- `javax.microedition.contactless.rf` per la comunicazione con i generici TAG RFID
- `javax.microedition.contactless.sc` per la comunicazione con le smart card definite dalla specifiche ISO 14443
- `javax.microedition.contactless.visual` per la lettura dei codici a barre

La compatibilità con le Contactless Communication API prevede che tutte le parti descritte dalle specifiche vengano incluse nell'implementazione ma viene lasciata la libertà di non implementare alcune funzionalità. In particolare risulta opzionale l'implementazione degli ultimi quattro package della lista di cui sopra mentre l'implementazione del primo, il javax.microedition.contactless, risulta obbligatoria. Poiché il tempo disponibile per ultimare questa tesi non è sufficiente per una trattazione dettagliata di tutte le funzionalità offerte dall'insieme completo delle API ci limiteremo ad illustrare unicamente quelle offerte da suddetto package.

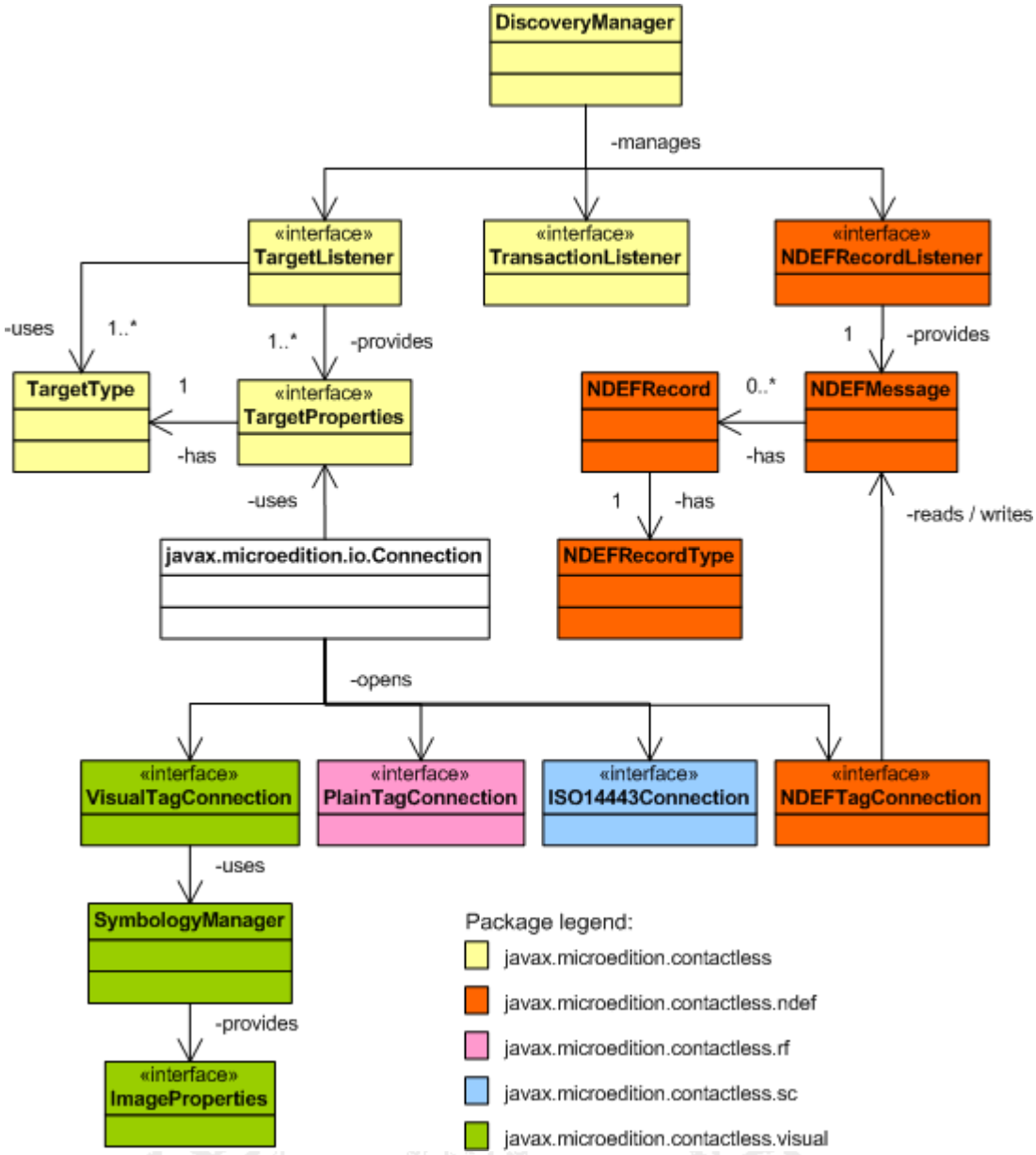
Come già detto in precedenza in questo package vengono definite le funzionalità comuni a tutti i tipi di Target supportati. L'entry point per l'utilizzo di queste API è rappresentato dalla rilevazione del TAG, funzionalità che viene messa a disposizione proprio in questo package. Viene, in particolare, fornito un meccanismo per cui risulta possibile, alle applicazioni che ne fanno richiesta, di ricevere notifica nel momento in cui viene rilevato un nuovo target. L'insieme delle interfacce e delle classi che compongono il package è illustrato tramite le tabelle mostrate di seguito. Per una descrizione più esaustiva di ciascuno degli elementi del package si rimanda alla documentazione Java fornita con le API.

Sommario delle Interfacce	
TagConnection	Rappresenta l'interfaccia che tutte le connessioni necessarie alla comunicazione con i target supportati dalle API devono estendere
TargetListener	Fornisce un meccanismo per il quale risulta possibile alle applicazioni di ricevere notifica nel momento in cui avviene la rilevazione di un nuovo target da parte del dispositivo
TargetProperties	Raccoglie le proprietà comuni a tutti i tipi di target supportati dalle API
TransactionListener	Fornisce all'applicazione notifica circa l'attività del *Secure Element* quando il dispositivo si trova in modalità *Card Emulation*

Sommario delle Classi	
DiscoveryManager	Fornisce i meccanismi per la rilevazione di TAG RFID
TargetType	Raccoglie le definizioni di tutti i tipi di target supportati

Sommario delle Eccezioni	
ContactlessException	Rappresenta l'eccezione che viene lanciata quando avviene un tentativo di operare un operazione non supportata

Diamo invece di seguito, tramite un diagramma, una panoramica delle relazioni che intercorrono tra gli elementi che compongono il package e che abbiamo precedentemente accennato.



Un applicazione che intende registrarsi presso il *DiscoveryManager* per ricevere notifica nel momento in cui avviene la rilevazione di un target deve necessariamente implementare l'interfaccia *TargetListener*. Suddetta registrazione avviene richiamando su di un istanza di *DiscoveryManager* il metodo *addTargetListener(TargetListener listener, TargetType targetType)* nel quale risulta necessario specificare l'istanza della classe alla quale si intende far ricevere notifica della rilevazione ed il tipo di target per il quale si desidera sia inviata la notifica. L'oggetto che implementa *TargetListener* deve fornire l'implementazione del metodo *targetDetected(TargetProperties[] properties)* il quale

viene richiamato nel momento in cui avviene la su citata rilevazione. Tramite il parametro *properties* di suddetto metodo di rende possibile l'accesso a tutte le proprietà riguardanti l'insieme dei target rilevati, compresi quelli necessari a stabilire una comunicazione e ad effettuare, quindi, le operazioni di lettura ed, eventualmente, scrittura col target. Quanto detto fino ad ora può essere illustrato tramite il seguente frammento di codice estratto da una MIDlet nel quale viene interrogato il *DiscoveryManager* per richiedere l'elenco dei tipi di target supportati dalla specifica implementazione delle API. Viene successivamente inoltrata richiesta di registrazione ad un istanza di *DiscoveryManager* per la ricezione della notifica di rilevazione di ciascuno di suddetti tipi di target.

```
import javax.microedition.contactless.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

public class TargetListenerDemo extends MIDlet implements TargetListener
{

      protected void startApp() throws MIDletStateChangeException {

            // Ottiene un elenco dei Targer supportati
            TargetType[] supportedTargetTypes =
                  DiscoveryManager.getSupportedTargetTypes();

            // Ottiene un istanza di DiscoveryManager
            DiscoveryManager dm = DiscoveryManager.getInstance();

            for( int i = 0; i < supportedTargetTypes.length; i ++ )
                  try {

                        // Imposta l'ascoltatore per il tipo di target
                        dm.addTargetListener( this,
                            supportedTargetTypes[ i ] );

                  } catch (IllegalStateException e) {

                        /*
                         * L'eccezione non viene mai sollevata in quanto
                         * la richiesta di registrazione per la notifica
                         * della rilevazione avviene una sola volta per
                         * ogni tipo di target supportato
                         */

                  } catch (ContactlessException e) {

                        /*
                         * L'eccezione non viene mai sollevata poichè
                         * facciamo richiesta di ricevere notifica di
                         * rilevazione dei target che sappiamo essere
                         * supportati
                         */
```

```
                }

        }

        public void targetDetected(TargetProperties[] arg0) {

                /*
                 * Per ciascuno dei target rilevati stampa a video i nomi dei
                 * tipi a cui corrisponde
                 */

                for( int i = 0; i < arg0.length; i ++ ) {

                        System.out.print( "Rilevato target di tipo " );

                        int n =  arg0[ i ].getTargetTypes().length;

                        for( int k = 0; k < n; k ++ )
                            System.out.print(
                                    arg0[ i ].getTargetTypes()[ k ].toString()
+ " " );

                System.out.println();

        }

    }

}
```

C.4 Le Bouncy Castle Cryptography API

Bouncy Castle rappresenta il nome di un progetto nato per lo sviluppo di un insieme di API per la realizzazione di operazioni crittografiche nei linguaggi di programmazione Java e C#. Anche in questo caso la mancanza di tempo ci impedisce di fare un esame dettagliato di tutte le operazioni che è possibile compiere tramite queste API. Ci limiteremo, infatti, a mostrare alcune delle funzionalità riguardanti la crittografia asimmetrica ed in particolare quelle relative alla creazione di una coppia di chiavi, alla cifratura e quelle per la generazione della firma digitale.

Illustriamo innanzitutto, tramite il seguente frammento di codice, il processo di creazione di una coppia di chiavi con particolare riferimento all'algoritmo RSA.

```
SecureRandom sr = new SecureRandom();
BigInteger pubExp = new BigInteger("10001", 16);
RSAKeyGenerationParameters RSAKeyGenPara =
      new RSAKeyGenerationParameters(pubExp, sr, 1024, 80);
RSAKeyPairGenerator RSAKeyPairGen = new RSAKeyPairGenerator();
RSAKeyPairGen.init(RSAKeyGenPara);
AsymmetricCipherKeyPair keyPair = RSAKeyPairGen.generateKeyPair();

RSAPrivateCrtKeyParameters RSAprivKey =
      (RSAPrivateCrtKeyParameters) keyPair.getPrivate();
RSAKeyParameters RSApubKey =
      (RSAKeyParameters) keyPair.getPublic();
```

Infine, mostriamo in che modo avviene, sempre tramite l'impiego dell'algoritmo RSA, la cifratura e la decifratura di un blocco di byte.

```
// Creazione di un array di byte da cifrare
long l = System.currentTimeMillis();
byte[] plainText = Long.toString( l ).getBytes();

// Cifratura
RSAEngine rsaEngine = new RSAEngine();
rsaEngine.init( true, RSAprivKey );
byte[] testoCifrato =
      rsaEngine.processBlock( plainText, 0, plainText.length );

// Decifratura
rsaEngine.init( false, RSApubKey );
byte[] plainText2 =
      rsaEngine.processBlock( testoCifrato, 0, testoCifrato.length );
```

Bibliografia

[1] P. Talone, G. Russo – 2008 – "RFId Fondamenti di una tecnologia silenziosamente pervasiva"

[2] D. Hunt, A. Puglia, M. Puglia – 2007 – "RFID A Guide to Radio Frequency Identification"

[3] Speciale RFID

http://www.ibiesse.it

[4] M. Zinato – Introduzione alla crittografia

http://sicurezza.html.it/guide/lezione/2378/introduzione-alla-crittografia

[5] M. Altese – 2006 – Nozioni base sulla sicurezza in Java

http://java.html.it/articoli/leggi/1696/nozioni-base-sulla-sicurezza-in-java

[6] U. Salsi – Crittografia

http://www.icosaedro.it/crittografia

[7] M. Yuan – Data security in mobile Java applications

http://www.javaworld.com/javaworld/jw-12-2002/jw-1220-wireless.html?page=7

[8] The Legion of the Bouncy Castle – Bouncy Castle 1.43 API Specification

http://www.bouncycastle.org

[9] G. Madlmayr – 2008 – NFC: Touch is the new click

http://blogs.forum.nokia.com/blog/gerald-madlmayrs-forum-nokia-blog

[10] C. Enrique Ortiz – 2008 – An Introduction to Near-Field Communication and the Contactless Communication API

http://java.sun.com/developer/technicalArticles/javame/nfc

[11] NFC Forum – 2006 – NFC Data Exchange Format, NFC Record Type Definition, NFC Text RTD Technical Specification, NFC URI RTD Technical Specification, NFC Smart Poster RTD Technical Specification

http://www.nfc-forum.org

[12] M. Carli – 2006 – MIDP 2.0 Book

http://www.massimocarli.it/site

[13] JSR 257 Expert Group – Contactless Communication API

http://jcp.org/aboutJava/communityprocess/final/jsr257